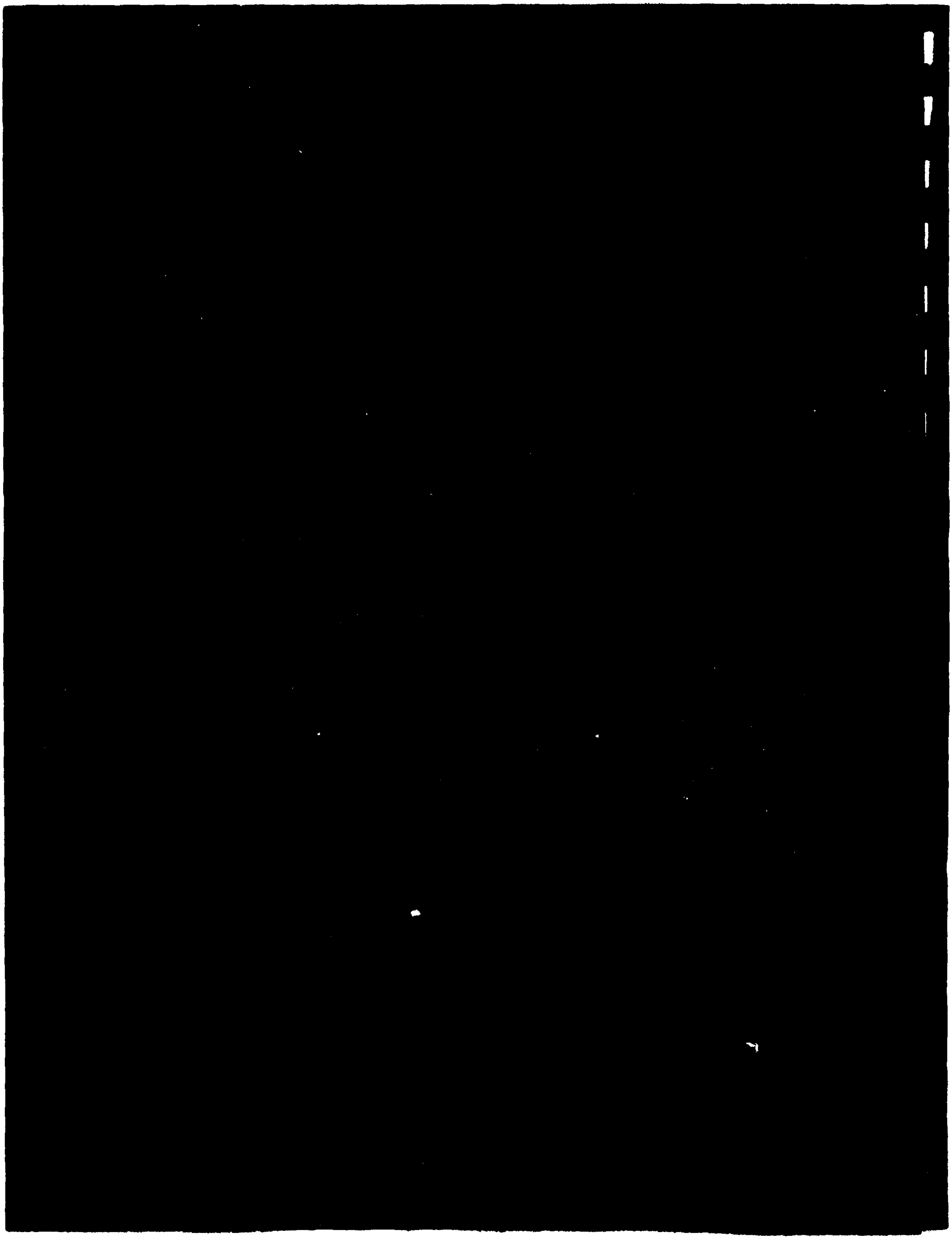
AD-A119 923   DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC   F/G 9/2
GPRIME: A GEOMETRIC LANGUAGE FOR FINITE ELEMENT MODELING PROGRA--ETC(U)
SEP 82   J M MCKEE, M E GOLDEN, D R WALLACE
UNCLASSIFIED   DTNSRDC-82/062   NL

1 of 2



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER DTNSRDC-82/062	2. GOVT ACCESSION NO. AD-A119 923	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) GPRIME: A GEOMETRIC LANGUAGE FOR FINITE ELEMENT MODELING PROGRAM MANUAL		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James M. McKee, Michael E. Golden and Dolores R. Wallace		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS David W. Taylor Naval Ship Research and Development Center Bethesda, Maryland 20084		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element 15325 Task Area SF-43-411-391 Work Unit 1808-009
11. CONTROLLING OFFICE NAME AND ADDRESS James J. Sejd Naval Sea Systems Command (03R2) Washington, D. C. 20362		12. REPORT DATE September 1982 13. NUMBER OF PAGES 123
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

GPRIME	Interactive Processing
Geometry Modeling	Finite Element
Computer Language	Structural Analysis
Computer Graphics	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The GPRIME geometric language is a computer language for describing and manipulating a wide variety of geometric shapes. The GPRIME language processor is a computer program that translates the geometric language description into a set of consistent mathematical definitions using B-spline approximation techniques. GPRIME application programs are computer programs, such as finite element data generators, that make use of the mathematical definitions created by the language processor.

(Continued on reverse side)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(Block 20 continued)

This manual is intended to be both a GPRIME primer for new users and a reference manual for experienced users. It describes the GPRIME geometric language, the GPRIME language processor, and recommended procedures for geometric modeling with GPRIME. With the GPRIME language it is simple to create mathematical models of complex geometry. Classical shapes are defined using straight-forward expressions, and more general curves and surfaces can be defined using digitized coordinate data.

The GPRIME processor uses interactive computer graphics for defining geometry and for aiding in visualizing the model being created. The processor creates a geometric data base which may be used for restarting the definition process at later sessions. This data base can also be accessed by other programs.

The GPRIME language was designed for use in creating finite element structural analysis models but can be applied wherever a description of geometry is required. Several finite element data generation programs now use GPRIME geometry for 2-D and 3-D modeling.

This manual defines the form and syntax of the GPRIME language and its command structure. Examples have been included to illustrate various concepts and procedures.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The GPRIME geometric language is a computer language for describing and manipulating a wide variety of geometric shapes. The GPRIME language processor is a computer program that translates the geometric language description into a set of consistent mathematical definitions using B-spline approximation techniques. GPRIME application programs are computer programs, such as finite element data generators, that make use of the mathematical definitions created by the language processor.

This manual is intended ~~to be both~~ a GPRIME primer, for new ~~users~~ and ~~a~~ reference ~~manual for~~ experienced users. It describes the GPRIME geometric language, the GPRIME language processor, and recommended procedures for geometric modeling with GPRIME. With the GPRIME language it is simple to create mathematical models of complex geometry. Classical shapes are defined using straight-forward expressions, and more general curves and surfaces can be defined using digitized coordinate data.

The GPRIME processor uses interactive computer graphics for defining geometry and for aiding in visualizing the model being created. The processor creates a geometric data base which may be used for restarting the definition process at later sessions. This data base can also be accessed by other programs.

The GPRIME language was designed for use in creating finite element structural analysis models but can be applied wherever a description of geometry is required. Several finite element data generation programs now use GPRIME geometry for 2-D and 3-D modeling.

This manual defines the form and syntax of the GPRIME language and its command structure. Examples have been included to illustrate various concepts and procedures.

ADMINISTRATIVE INFORMATION

The work reported herein was performed as part of Program Element 15325, Task 15325, Task Area SF-43-411-391 under Work Unit 1808-009 at the David W. Taylor Naval Ship Research and Development Center.

INTRODUCTION

As structural engineers are well aware, the creation of a correct finite element model is no mean feat. The large amount of data required, the many details to be considered, and the hundreds of dollars worth of computer processing time required make the finite element analysis of even a small

problem a significant task. Preparation of the data is by far the most costly and error prone segment of the analysis. Almost any conceivable structure, from miniature valves to entire ships, may be analyzed by the finite element method. This extreme generality of size and shape has frustrated attempts to create an effective automatic finite element data generator. Initially, we viewed the development of a geometric language as a general solution to the problem of automating the finite element modeling process. As we began to formulate such a language there seemed to be many other applications which could benefit from this type of consistent geometric representation. We called this language GPRIME.

The basic assumption of the GPRIME approach is that, if a systematic and accurate mathematical model of the geometry of a structure could be stored in the computer, the task of creating general finite element data generators would be greatly simplified. The advent of low cost, interactive computer graphics makes the GPRIME approach even more attractive, since problem areas in an automatically generated model, can be easily resolved by manual methods at the interactive screen. Even the generation of a finite element grid at the junction of intersecting cylinders, which is quite difficult to automate, becomes a simple task with a little interactive help from the program user.

At the time the GPRIME project was initiated, several programs were available for geometric modeling. Automatic drafting programs represent one type of geometric capability which can save significant amounts of time in creating and updating engineering drawings. Because these programs were direct computerizations of manual drafting processes, they required manual interpretation of conventions and symbols to construct the geometry and are thus not very suitable for automatic data generation. A second type of geometric capability was designed for programming numerically-controlled tool operations. Although these programs produce a more complete picture of the structural geometry than the automatic drafting programs, geometric information is usually available only in the form of cutter paths and is not very appropriate for finite element modeling.

The GPRIME project team designed a geometric description capability which could accurately model both classical geometric shapes and arbitrary curves and surfaces defined by collections of digitized points. The capability was

to be graphically oriented (like the automatic drafting programs), easy to learn, and easy for users to maintain their proficency in its use (an improvement over the drafting programs). The resulting GPRIME language can be used passively or in an interactive graphics mode, can be stopped and restarted from any point, and has interactive "help" features to refresh the user's memory on any aspect of the modeling process. It also has several general purpose data generators which operate on the geometry defined, and it permits direct access to the geometric model for user-written, special purpose generators. Our implementation of this language was greatly simplified by using cubic B-spline functions as the internal mathematically representation of all curves and surfaces.

GPRIME development began along three parallel paths: (1) development of a B-spline mathematics capability, which resulted in a library of B-spline subroutines,[1*] (2) development of the language processor,[2] and (3) development of general purpose data generators.[3] Each of these efforts is now mature enough to allow creation of finite element models in the manner prescribed at the outset of the project. Early users of GPRIME have been able to produce complex models like the propeller shown in Figure 1 and the piping joint shown in Figure 2.

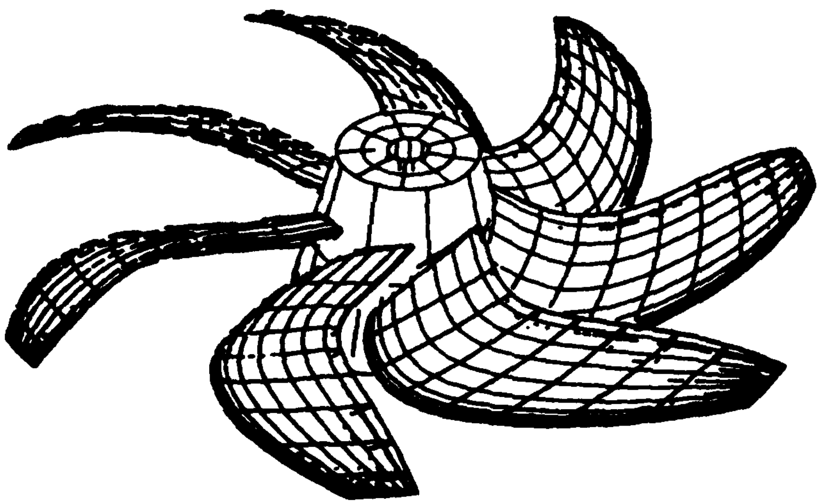


Figure 1 - GPRIME Propeller Model

*A complete listing of references is given on page 111.

In the development of the GPRIME processor many features have been added which were not originally specified (e.g., hidden line removal). Early users have also contributed many suggestions which have been incorporated into the language and we hope that user feedback will continue to influence the further development of GPRIME.

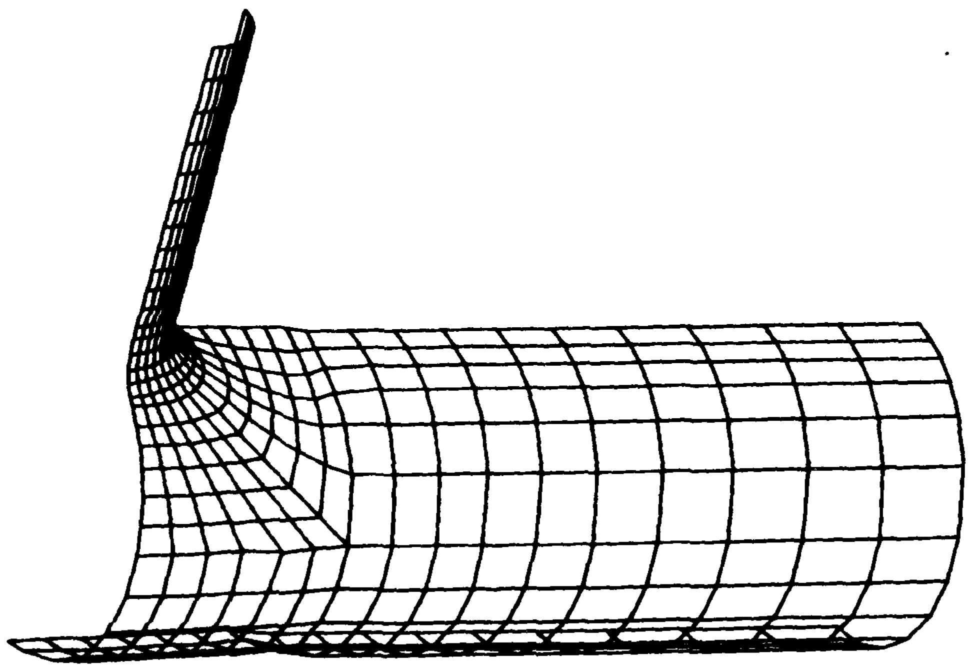


Figure 2 - GPRIME Generated Finite Element Model of Pipe Tee

The GPRIME processor was developed on CDC 6000 series computers and is now running on several such machines under the NOS/BE and NOS operating systems. Although other terminals have been used from time to time, GPRIME'S interactive graphics capability is primarily targeted for Tektronix storage-tube-type terminals. GPRIME is being converted to run on medium to large minicomputers and it is expected that conversion to other computers and other graphic devices will require minimal programming effort.

While GPRIME was being developed, the automatic drafting and numerically-controlled tools programs were also evolving. Current versions of the automatic drafting programs incorporate many numerically-controlled tools features and more pure geometric and finite element modeling features as well. The numerically-controlled tools programs have evolved with broader geometric definition capabilities. The merging of GPRIME'S capability with these other programs could produce a powerful design and analysis system.

This manual is intended to be both a GPRIME primer for the new user and a reference manual for the experienced user. We have structured it so as to first expose the reader to a simple modeling problem and a typical GPRIME solution to that problem. Then the form and syntax of GPRIME geometric definitions are covered in detail. Following that, various commands are described that influence the GPRIME environment (graphics, restarting user help, etc.). The definition and command sections have their own indexes for ready reference. At the end of the manual there are discussions on the independent use of GPRIME'S geometry and on computer considerations for running the GPRIME processor.

A FEW BASIC TOOLS

The reader will be introduced to the GPRIME language with a simple example, preceded by a brief discussion of the form and syntax of some key elements. The basic syntax of GPRIME language statements requires the user to provide a list of keywords and parameters which are separated by commas (to improve readability, blanks may be inserted both at the beginning of statements and around any comma).

VARIABLES

A GPRIME variable is established for each geometric item defined using the GPRIME language. GPRIME variables are variable in the sense that they are symbols which may be defined in terms of other geometric variables. They may be used to define other geometric variables and may take on new values as conditions change. Typically, variable names are assigned by the person using the program. In a few cases, the GPRIME processor assigns variable names for the user.

Variable names begin with one or two alphabetic characters which identify the broad geometric category to which the item belongs (e.g., "S" for surfaces). The rest of the variable name is an integer number chosen to identify that particular item. A point could be named P17, a curve, C123, and a surface, S300.

DEFINING POINTS

GPRIME permits geometric data to be entered in 3-D rectangular, cylindrical, and or spherical coordinate systems. Once entered, all definitions are transformed to a base 3-D rectangular system.

A point at the origin of GPRIME'S base coordinate system, to be referenced as P0, can be defined by the statement:

```
P0, POINT, 0, 0, 0
```

A point may also be defined as part of some other definition by enclosing the coordinates of that point in parentheses:

```
..., P5, P7, (0, 0, 0), P13, ...
```

This is one situation in which the GPRIME processor assigns the variable name (e.g., P503).

DEFINING A CIRCULAR ARC

A three-dimensional circular arc (all GPRIME geometry is considered to be three-dimensional) can be defined by giving the coordinates of the center of the circle and the coordinates of the end points of the arc. The statement:

```
C2, ARC, P1, P3, P5, CENTER
```

defines a curve C2 to be such an arc, where P1 is the point defining the center of the circle and P3 and P5 are end points of the arc. Similar arcs can be defined without explicitly creating the point variables, as in the statement:

```
C3, ARC, (0,0,0), (0,1.5,0), (0,0,1.5), CENTER
```

DEFINING A PLANAR SURFACE

A planar rectangular region perpendicular to an axis of the base coordinate system represents one type of surface which can be defined with GPRIME.

A z=0 planar surface S1 is defined by the statement:

S1, PLANE, 0., -1., 1., -2., 2., Z

This plane extends from x=-1 to x=+1 and from y=-2 to y=+2.

DEFINING A RULED SURFACE

A surface formed by straight line generators connecting corresponding points on a pair of 3-D arcs is referred to as a ruled surface. A ruled surface S1 can be defined in the GPRIME language by the statement:

S1, SURFACE, C1, C2

where C1 and C2 are 3-D arcs. Internally, all GPRIME curves are defined parametrically, have finite length, and are normalized so that a parameter value of zero is associated with the beginning of each curve and a parameter value of one is associated with the end of each curve. (This does mean that a closed curve, such as a circle, will have pseudo beginning and ending points which coincide.) For GPRIME ruled surfaces, straight line generators connect points which correspond parametrically on the two defining arcs.

DEFINING A RIGHT CIRCULAR CYLINDER

Internally, all GPRIME surfaces are defined parametrically and have finite extent, analogous to the curve definitions. Surface definitions are functions of two parameters and are normalized so that the parameters range between zero and one over the surface. Closed surfaces, like a right circular cylinder, also have a pseudo-seam where the surface "begins" and "ends". A right circular cylinder S2 can be defined by the statement:

S2, CYLINDER, P1, P2, P3

where P1 is a point that defines the end of the axis of the cylinder at the base, P2 is a point that defines the opposite end of the axis, and P3 is a point on the circumference of the cylinder at the base. (The pseudo-seam will pass through P3.)

DEFINING A CURVE OF INTERSECTION BETWEEN TWO SURFACES

A curve C3, which is the curve of intersection between two GPRIME surfaces S1 and S2, can be defined by the statement:

C3, INTERSECTION, S1, S2

The type of curves resulting from such intersections can be quite general. However, the intersections must be unique, that is, if the intersection results in two or more disjoint arcs, the definition will be unacceptable (probably one surface will have to be subdivided).

REPLICATION OF GEOMETRY

Sometimes it is useful to have two or more GPRIME variables associated with the same geometric construction. The statement:

S10, S1

defines a surface S10 to be identical to the surface S1.

DEFINING A GEOMETRIC TRANSFORMATION

A GPRIME geometric construction can be defined as the rotation and translation of any other geometric construction by way of an orientation transformation. Orientation transformation OT4 is defined by the statement:

OT4, ORIENTATION, 22.5, 0., 90., P6

where the three parameters following the key word "ORIENTATION" specify angular rotations about the base x-axis, y-axis, and z-axis, respectively (the rotation angles are in degrees, the positive direction to be determined by the "right hand rule", and are to be applied in that order), and the coordinates of the point P6 specify translations in the three coordinate directions. Orientation transformations are applied by appending the orientation transformation variable name as the last parameter of a definition statement. For example:

S1, SURFACE, C1, C2, OT4 and

S5, S1, OT4

both define transformed surfaces that are reoriented with respect to some reference surface.

Assuming that the reader is now familiar with the set of basic GPRIME tools, we will discuss a simple example.

AN EXAMPLE

The following GPRIME definition statements can be used to construct a model of the junction of cylinders shown in Figure 3.

```
C1, ARC, (-10,0,0), (-10,10,0), (-10,0,10), CENTER
OT1, ORIENT, 45, 0, 0, (0,0,0)
C2, C1, OT1
OT2, ORIENT, 0, 0, 0, (20,0,0)
C3, C2, OT2
S1, SURFACE, C2,C3
S2, PLANE, 0, -10, 10, -10,10, Z
S3, CYLINDER,(-10,0,-4.364),(10,0,4.364),(-10,2,-4.364)
C4, INTERSECT, S2, S3
OT3, ORIENT, 0, 0, 0, (0,0,15)
C5, C4, OT3
S4, SURFACE, C4, C5
C6, INTERSECT, S1, S4
S5, SURFACE, C6, C5
```

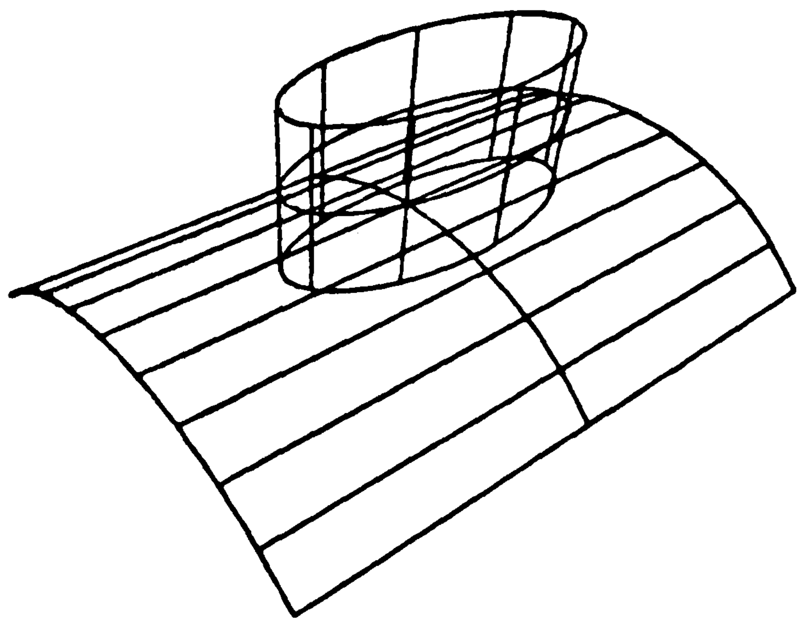


Figure 3 - Junction of Circular and Elliptic Cylinders

With the aid of the GPRIME tools described in the preceding section, the reader should be able to interpret the role of each statement and to identify

the elements of construction in the GPRIME graphics display shown in Figure 4.

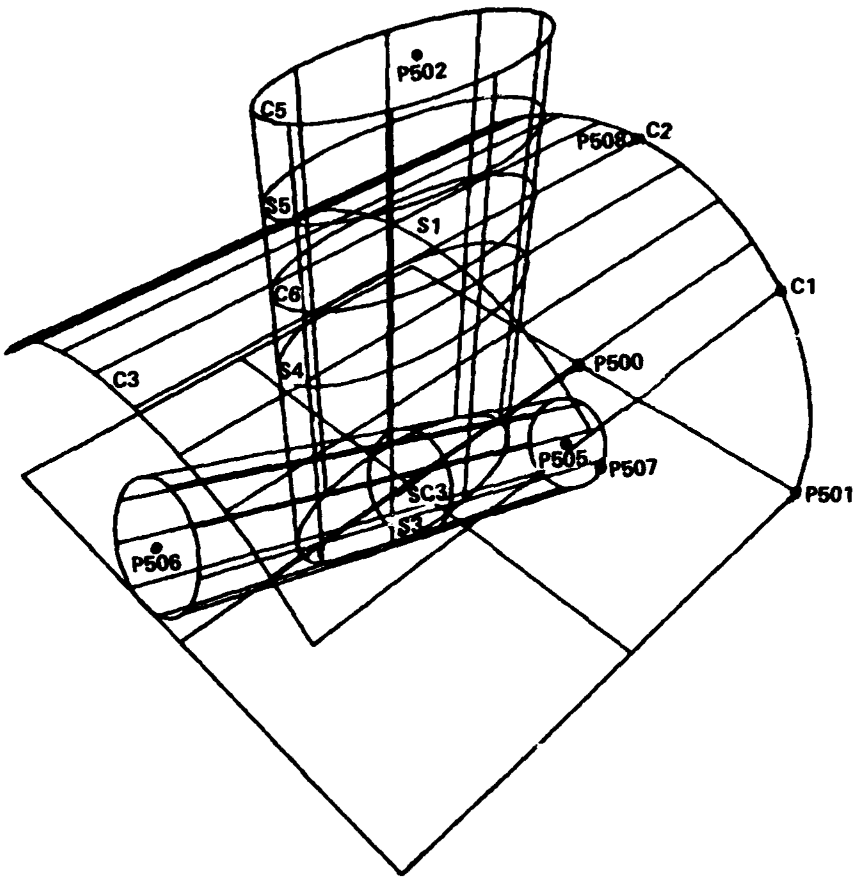


Figure 4 - Graphics Display for Construction of GPRIME Model

This example contains few, if any, subtleties; however, the reader may wish to compare his interpretation of the definitions with the following commentary:

C1 is defined to be a 90-degree circular arc which lies in the x = -10 plane and appears at the right in Figure 4.

OT1 defines a 45-degree counterclockwise rotation about the x-axis.

C2 is defined to be a curve similar to C1 but rotated by 45 degrees.

OT2 defines a translation of 20 units in the positive -x direction.

C3 is defined to be a curve similar to C2 but translated by 20 units (at the left in Figure 4).

S1 is defined to be a ruled surface between curves C2 and C3 (i.e., a cylindrical arch).

S2 is defined to be a z=0 planar region which is a square measuring 20 units on a side and centered at the origin.

S3 is defined to be a right circular cylinder, with a radius of two units, which passes obliquely through the plane S2.

C4 is an elliptical curve defined as the intersection of the cylinder S3 and the plane S2.

OT3 defines a translation of fifteen units in the positive-z direction (upward in Figure 4).

C5 is defined to be a curve similar to the ellipse C4 but translated upward by 15 units.

S4 is defined to be a ruled surface between curves C4 and C5 (i.e., a right elliptical cylinder).

C6 is defined to be the curve of intersection between the cylindrical arch S1 and the right elliptical cylinder S4.

S5 is defined to be the ruled surface between the junction curve C6 and the ellipse C5.

This example highlights the primary function of the GPRIME language-geometric modeling. Various program control and supporting functions of the GPRIME processor are illustrated by the following set of commands:

(1) HIDE, ALL
(2) PLOT, S1, S5
(3) IDENTIFY, ON
(4) VNT, ON; VDT, ON
(5) LABELS, OFF
(6) VIEW, EYE, 30, 30, 30, CLIP, 100
(7) STATUS

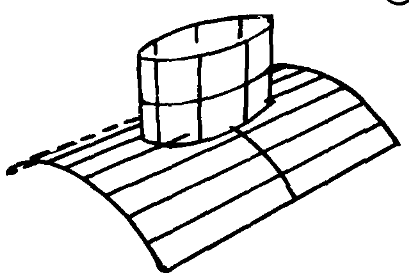
Applying these commands in sequence to the geometry defined previously (as seen in Figure 4) produces the display shown in Figure 5.

15.07.39 05/21/80 ← A
TIME LEFT – 67 SECONDS

P500
P501
P502
C1
P503
OT1
C2
P504
OT2
C3
S1
S2
P505
P506
P507
S3
C4
P508
OT3
C5
S4
C6
S5

B

C3, C2, OT2
S1, SURFACE, C2, C3
S2, PLANE, 0, –10, 10, –10, 10, Z
P505, POINT, –10, 0, –4.364
P506, POINT, 10, 0, 4.364
P507, POINT, –10, 2, –4.364
S3, CYLINDER, P505, P506, P507
C4, INTERSECT, S2, S3
P508, POINT, 0, 0, 15
OT3, ORIENT, 0, 0, 0, P508
C5, C4, OT3
S4, SURFACE, C4, C5
C ↗ C6, INTERSECT, S1, S4
S5, SURFACE, C6, C5



EYE			
	30.000	30.000	30.000
ETOS		7.500	
TRANSLATE			
	0.000	0.000	0.000
SCALING		1.000	
ANGLES			
	0.000	0.000	0.000
D → AXES	OFF		
DEBUG	OFF		
DRAW	OFF		
DUMP	OFF		
LABEL	OFF		
PRINT	OFF		
TIME	OFF		
TRACE	OFF		
UDT	ON		
UNT	ON		
POINTS	ON		
CURVES	ON		
SURFACES	ON		
HIDE, ALL			
VARIABLES TO PLOT			
S1	S5		
TOLERANCE		4	
FUNCTIONS		8	

E

STATUS

Figure 5 - Modified Display of GPRIME Model

Individually, the commands have the following effects:

(1) HIDE is used to specify a subset of the geometry which is not to be displayed at the screen. In this case, nothing is to be displayed.

(2) PLOT is used to specify a subset of the geometry which is to be displayed at the screen. This command indicates that surfaces S1 and S5 are to be displayed.

(3) IDENTIFY is used to request that the current time and date be displayed at the top of the screen (Figure 5, Item A). The TIME LEFT message, which appears below the time and date, is always displayed and indicates the computer time remaining for GPRIME processing at this session (in seconds). If time runs low, the session may be ended and the processing restarted with a reset time allocation. (This time restriction is a function of the host computer's interactive operating system and will vary with each installation.)

(4) Several GPRIME statements may be entered on one type-in line by separating the statements with semicolons (;). The Variable Name Table, or VNT, is a list of all GPRIME variables currently defined and appears at the left of the screen (Figure 5, Item B). The Variable Definition Table, or VDT, is a list of the GPRIME definitions entered during the session and currently active. This table appears at the top of the screen (Figure 5, Item C). These commands request that the two tables be displayed.

(5) Each viewable element of GPRIME geometry may have its associated variable name displayed on or near that element. This LABEL command requests that no variable names be displayed.

(6) The VIEW command is used to specify parameters for GPRIME'S 3-D perspective plotting. This command is used to request that the display be redrawn (VIEW always redraws the display), that the observation point (the viewer's EYE) is to be moved to the point (30,30,30) in the base coordinate system, and that the plotting is to be done with hidden lines removed (CLIP). The integer "100", which follows the key word CLIP, is used to indicate a moderate level of refinement in the hidden line processing (Details will be given later).

(7) This command requests that the current STATUS of all screen and projection parameters be displayed at the right of the screen (Figure 5, Item D).

The display of the type-in line near the bottom of the screen (Figure 5, Item E) and accompanying audible "beep" are the GPRIME processor's indication that it has finished processing a statement and is ready to accept additional entries.

This example includes many of the most frequently used GPRIME constructions. Detailed descriptions of these and all other GPRIME facilities are given in the remainder of this report.

GEOMETRIC DEFINITION

Creation of a geometric model with the GPRIME language requires little more than the logical decomposition of a geometric shape into simple components and the coding of component descriptions into GPRIME definition statements. For most shapes, at least for those shapes routinely produced by a draftsman, the user need not be aware of GPRIME's underlying mathematical formulations. Some knowledge of the B-spline mathematics used by GPRIME will be required for successful modeling of complex shapes and of curves and surfaces described by digitized data. We will explore the necessary B-spline concepts later in this report (See sections: B-SPLINE FUNCTIONS, DATA FITTING AND THE USE OF DIGITIZED DATA, and FINDING INTERSECTIONS).

In the sections that follow, each type of definition will be described in considerable detail. These details are intended as a reference for practicing GPRIME users and can be passed over until the reader needs to apply a specific feature. Most of these details are also available from GPRIME's interactive HELP command.

From a language point of view, definition statements are the essence of GPRIME. Each geometric element is defined via a GPRIME definition statement. Certain GPRIME commands can be used to define several geometric elements, but for each element defined by a multiple definition command, a corresponding definition statement is generated internally.

All definition statements are saved in the User Master File (UMF) and can be displayed at any time. These statements will be referenced during automatic updating of dependent model geometry that is invoked whenever defining variables are changed. A file of all current definition statements can be obtained for archival backup of the model.

During an interactive GPRIME session the user may obtain summary information for definition statements by invoking the HELP command. The "HELP,DEFINITION" command will produce a listing of all possible definition type keys and the "HELP, DEFINITION, <definition type key>" command will produce a short description of the specified type of definition statement.

VARIABLES

A GPRIME variable is established for each geometric item defined using the GPRIME language. GPRIME variables are variable in the sense that they are symbols which may be defined in terms of other geometric variables. They may be used to define other geometric variables and may take on new values as conditions change. Typically, variable names are assigned by the person using the program. Certain implied definitions, and definitions produced by multiple definition commands, will have variable names assigned by the GPRIME processor.

Variable names begin with one or two alphabetic characters. These characters identify the broad geometric category to which the item belongs. The rest of the variable name is an integer number chosen to identify that particular item. Somewhat more formally a GPRIME variable name has the following characteristics (where ":=" is read "is defined to be"):

<variable name>	:= <variable type key> <integer number>	
<variable type key>	:= SC	for scalar variables
	:= P	for point and vector variables
	:= C	for curve variables
	:= S	for surface variables
	:= G	for group variables
	:= OT	for transformation variables
	:= BP	for B-spline parameter variables

When the GPRIME processor assigns a variable name, the integer portion of the name is chosen from a sequence of consecutive integers, beginning with 500. If a variable name with the chosen number is already in use, the program will continue to choose numbers until a unique name is obtained. The maximum

length of a variable name depends on the word size of the host computer. On CDC 6000 series computers variable names are limited to five characters.

Once defined, variables and their associated definition statements can be removed from the model using the DELETE command, and variables can be changed by modifying their definition statements with the CHANGE command (see COMMANDS section).

DEFINITION STATEMENT SYNTAX

Each GPRIME variable has an associated definition statement. Definition statements consist of the dependent variable name (the variable to be defined), a key word which specifies the type of definition, a predicate consisting of independent variable names and parameters, and an optional transformation. A definition statement has the following characteristics:

<dependent variable> := <definition type key>, <predicate>,
where: [,<transformation variable>]

<definition type key>:= <alphanumeric string> (defined in following sections, e.g., "POINT")

<predicate> := <independent variable>
:= <parameter>
:= <parameter>, <predicate>
:= <predicate>, <parameter>
:= <independent variable>, <predicate>
:= <predicate>, <independent variable>

```
<parameter>            := <integer number string>
                       := <real number string>
                       := <alphanumeric string>
```

The definition type key (e.g., point, curve, intersection) is used to select a particular form of definition. Type keys are given for each statement in the descriptions that follow. An alphabetical cross reference to definition type keys is given at the end of the GEOMETRIC DEFINITION section.

Elements of definition statements must be separated by commas. Blank characters can be inserted between elements as desired. Default values are obtained, where appropriate, by truncating the definition or by specifying null or blank elements (delimited by commas).

Numeric parameters must conform to FORTRAN rules for numeric constants. Note that integer values are acceptable as real parameters. Alphanumeric parameters can be any contiguous string of characters which does not contain blank or punctuation characters (commas, parentheses, and semicolons), or any string of characters delimited by dollar signs (e.g., (X,Y,Z)).

The maximum recognized length of a parameter is governed by the word size of the host computer (ten characters for CDC 6000 series computers). All definition type keys and alphanumeric parameters can be abbreviated to the first three characters of the string. (The only exception to this abbreviation rule occurs in the execution of macro commands where macro names must be entered exactly as they are defined by the user.)

Interactive graphic definitions require the use of a graphic input device at the user's terminal (light pen, tracking cross, cross-hair cursor, etc.). Any graphic input device that is active at the user's terminal will be selected when the alphanumeric parameter "TC" (Tracking Cross) is specified.

In the descriptions of the GPRIME definition statements that follow:

(1) Variable names are indicated by an upper-case type key followed by a lower-case identifying character (e.g., Pi, SCj).

(2) Definition type keys and required alphanumeric parameters are given in upper-case characters (e.g., POINT, CIRCUMFERENCE).

(3) Integer parameters are indicated by a lower-case "k" followed by an identifying integer (e.g., k1, k5).

(4) Real number parameters are indicated by a lower case "r" followed by an identifying integer (e.g., r1, r7).

(5) Alphanumeric parameters are indicated by a lower case "a" followed by an identifying integer (e.g., a1, a21).

In this format, upper-case characters indicate required text (or a suitable abbreviation thereof) and lower-case characters indicate values that are to be supplied by the user.

SCALAR DEFINITION

Scalar quantities often appear as constants in GPRIME definition statements, but scalar quantities are not usually established as formal variables.

When the need arises there are several ways for the GPRIME user to define scalar variables. (No GPRIME commands create scalar variables.)

1. Basic Scalar Definition--SCALAR

Function: To define a scalar variable and to assign a value to that variable.

Format: SCi, SCALAR, r1

Note: The real value to be assigned to r1 must fall within the limits for real-number representation on the host computer.

<<<:>>>

2. Scalar Defined as a Component of a Point or Vector--SCALAR

Function: To define a scalar variable and to assign a value to that variable from a component of point coordinates or a vector.

Format: SCi, SCALAR, <a1><k1>

Note : <a1> := "X" for extracting the x- (or first) component
:= "Y" for extracting the y- (or second) component
:= "Z" for extracting the z- (or third) component
<k1> := identifying integer number of the point or vector variable, Pk1, from which the component value is to be extracted. For example, SC3,SCALAR,Y13 will define a scalar as the Y-coordinate of the point P13.

<<<:>>>

3. Scalar Defined as the Distance Between Two Points--SCALAR

Function: To define a scalar variable and to assign a value to that variable which is the distance between two specified points.

Format: SCi, SCALAR, Pj, Pk, DISTANCE

Note: No restrictions.

<<<:>>>

4. Scalar Defined Graphically--SCALAR

Function: To define a scalar variable and to assign a value to that variable from a graphic input device.

Format: SCi, SCALAR, TC, <a1>

(a) The component of the graphically-defined point is selected as follows:

Notes: <a1> := "X" for obtaining the x-component of the graphic input point

:= "Y" for obtaining the y-component of the graphic input point

:= "Z" for obtaining the z-component of graphic input point

(b) The graphic input points used for these definitions lie in a viewing plane which passes through the viewing origin (see VIEW command) and is parallel to the plane of the viewing screen. For a particular view the z-components of all such points will be the same (i.e., the distance from the viewer's eye to the viewing origin).

POINT DEFINITION

In addition to assigning coordinate values directly, GPRIME permits points to be defined graphically, to be defined using digitized input, and to be defined using the results of curve and surface intersection processes. Several GPRIME commands can also be used to define points, including the "POINTS" command and the "GENERATE, ARC" command. The GGEN interactive data generator can also be used to define points.

1. Basic Point Definition--POINT

Function: To define a point or vector variable and assign a value to that variable.

Format: Pi, POINT, r1, r2, r3 [,a1]

Notes: (a) r1, r2, and r3 are the coordinates of the point being defined.

(b) a1 := "RECTANGULAR" or omitted to define a point in a Cartesian coordinate system.

:= "CYLINDRICAL" to define a point in a cylindrical coordinate system.

:= "SPHERICAL" to define a point in a spherical coordinate system.

Cylindrical and spherical coordinate systems are oriented, relative to the base rectangular system, as shown in Figure 6. All GPRIME coordinate systems have a common origin, "O" in Figure 6. The interpretation of the coordinates depends on a1 as follows:

a1	r1	r2	r3
RECTANGULAR	x	y	z
CYLINDRICAL	r	θ_c (degrees)	z
SPHERICAL	ρ	ϕ (degrees)	θ_s (degrees)

<<<:>>>

2. *In-Line Point Definition*--(x,y,z)

Function: To define a point or vector variable, within a definition statement that requires a variable, and to assign a Cartesian coordinate value to that variable.

Format: ...(r1, r2, r3)...

Notes: (a) r1, r2, r3 are the Cartesian coordinates of the point being defined.

(b) The GPRIME processor will assign a unique variable name to each point variable defined "in-line".

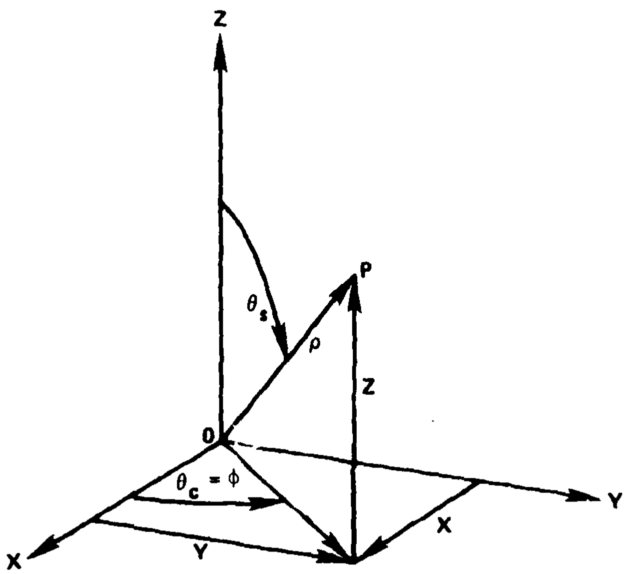


Figure 6 - Coordinate System Orientation

3. Point Definition by Graphic Input--POINT

Function: To define a point or vector variable and to assign values to that variable using a graphic input device.

Format: Pi, POINT [,a1], TC

Notes: (a) a1 is the name of an optional reference curve or surface. If present, the point will be defined as the intersection of the reference element with the line of sight that passes through the graphic input point. After the point has been defined, the form of the definition will be changed to parametric form (Pi, POINT, a1, r1 [,r2]) for future reference.

(b) If a1 is omitted, the point will be defined as the intersection of the viewing plane (which passes through the viewing origin and is parallel to the plane of the viewing screen--see VIEW command) with the line of sight that passes through the graphic input point. After the point has been defined, the form of the definition will be changed to the basic form (Pi, POINT, r1, r2, r3) for future reference.

(c) If a graphic input point is selected which does not lie on the reference element but is very close to that reference element, the point will be defined as the point on the reference element that is closest to the line of sight through the graphic input point. This feature may be used to precisely position a point on the "edge" of a surface or at the "end" of a curve.

d) For automatic repetition of this form of definition, see the POINTS command.

<<<:>>>

4. Point Defined From a File of Digitized Data--POINT

Function: To define a point or vector variable and to assign Cartesian coordinate values to that variable from a file of digitized data.

Format: Pi, POINT, k1

Note: The integer constant, k1, is used to specify the record on the digitized data input file (file name SPDATA) from which the coodinates of the point are to be obtained. If k1 = 0, the coordinates of the point will be

obtained from the next record on the primary input file (e.g., keyboard input). For format information see the section: DATA FITTING AND THE 'SE OF DIGITIZED DATA.

<<<:>>>

5. Point Defined by Parametric Reference --POINT

Function: To define a point or vector variable and to assign coordinate values that are defined by the parametric coordinates of a point on a curve or surface.

Format: Pi, POINT, a1, r1 [,r2]

Notes: (a) The alphanumeric constant a1 must be the name of a curve or surface variable (Cj or Sk).

(b) A single real constant r1 is the parametric coordinate used to select a point on the reference curve Cj, using GPRIME's internal (B-spline) definition form.

(c) A pair of real constants r1 and r2 are parametric coordinates used to select a point on the reference surface Sk using GPRIME's internal (B-spline) definition form.

(d) The section B-SPLINE FUNCTIONS should be consulted for further details on GPRIME's internal form of curve and surface representation.

<<<:>>>

6. Point Defined by Intersecting Three Surfaces--INTERSECTION

Function: To determine a point of intersection of three GPRIME surfaces, to define a point or vector variable, and to set the value of that variable to the coordinates of the point found.

Format: Pi, INTERSECTION, Sj, Sk, Sl [,Pm]

Notes: (a) There are no restrictions on the surfaces used in this form of point definition (S1, S2, S3); however, the user is referred to the section FINDING INTERSECTIONS for additional information on GPRIME's intersection process.

(b) The optional point Pm is used as a "guide point" to select among several possible points of intersection. In the multiple intersection case, the point of intersection which is closest to the point Pm will be selected as

the desired value. When a unique intersection is known to exist, the use of the auxiliary point Pm is unnecessary and very inefficient.

<<<:>>>

7. Point Defined by a Curve/Surface Intersection--INTERSECTION

Function: To determine a point of intersection of a curve and a surface, to define a point or vector variable, and to set the value of that variable to the point found.

Format: Pi, INTERSECTION, Cj, Sk [,Pl]

Notes: (a) There are no restrictions on the curve and surface used in this form of point definition (Cj and Sk); however, the user is referred to the section FINDING INTERSECTIONS for additional information on GPRIME's intersection process.

(b) The order of the curve and surface variables may be interchanged, if desired.

(c) The optional point Pl is used as a "guide point" to select among several possible points of intersection. If there are multiple intersections, the point of intersection which is closest to the point Pl will be selected as the desired value. When a unique intersection is known to exist, the use of the auxiliary point Pl is unnecessary and very inefficient.

<<<:>>>

8. Point Defined as the Intersection of Two curves--INTERSECTION

Function: To determine a point of intersection of two curves, to define a point or vector variable, and to set the value of that variable to the coordinates of the point found.

Format: Pi, INTERSECTION, Cj, Ck [,P]

Notes: (a) There are no restrictions on the curves used in this form of point definition (Cj and Ck); however, the reader is referred to the section FINDING INTERSECTIONS for additional information on GPRIME's intersection process.

(b) It is often impossible to know in advance whether two fitted three-dimensional curves have a point of intersection. This form of point definition assumes that a point of intersection has been found if the distance between the two curves is very small (governed by the TOLERANCE command). The

value of the variable will be an average calculated from the closest points from each curve.

(c) The optional point P1 is used as a "guide point" to select among several possible points of intersection. If there are multiple intersections, the point of intersection which is closest to the point P1 will be selected as the desired value. When a unique intersection is known to exist, the use of the auxiliary point P1 is unnecessary and very inefficient.

CURVE DEFINITION

Classical curves can be defined in more-or-less conventional mathematical terminology with the GPRIME language. There are several methods for defining more general curves, including a two-surface intersection capability and the use of digitized data. All GPRIME curves are considered to have finite extent; however, linearly extrapolated values can be obtained outside the range of original definition, when necessary. There are no GPRIME <u>commands</u> which automatically define curves.

Internally, all GPRIME curves are represented by B-spline functions which are used to fit the definitions specified. This form of internal representation is very broad and can be easily applied to almost any new situation. The internal curve representations are stored as polygons of point data (this form is also required for defining curves via digitized data). Although a knowledge of GPRIME's B-spline conventions is not necessary for basic problems, it can greatly enhance the range of potential applications for GPRIME's curve representation capability. For further information the reader may refer to the section: B-SPLINE FUNCTIONS.

1. Straight Line Connecting Two Points--LINE

Function: To establish a curve variable defined to be a straight line connecting two points.

Format: Ci, LINE, Pj, Pk

Note: No restrictions.

<<<:>>>

2. Circle--CIRCLE

Function: To establish a curve variable defined by specifying its

center and two points on its circumference, or by specifying three points on its circumference.

Format: ci, CIRCLE, Pj, Pk, Pl, al [,NOCHECK]

Notes: (a) The point Pj will be assumed to define the center of the circle if the alphanumeric parameter al is CENTER. In this case the points Pk and Pl must be equidistant from the center point Pj. If the word NOCHECK is included as the last parameter, the radius of the circle will be defined as the distance between the points Pj and Pk, and the plane of the circle will contain the point Pl, but no check will be made to verify that the point Pl lies on the circle thus defined.

(b) If the alphanumeric parameter al is CIRCUMFERENCE, all three points will be assumed to lie on the circumference of the circle.

(c) The three points Pj, Pk, and Pl must not be collinear.

<<<:>>>

3. Circular Arc--ARC

Function: To establish a curve variable that is a circular arc defined by specifying its center and end points, or by specifying three points on the arc, the first and last of which are its endpoints.

Format: Ci, ARC, Pj, Pk, Pl, al [,NOCHECK]

Notes: (a) The point Pj will be assumed to define the center of the circular arc if the alphanumeric parameter al is CENTER. The points Pk and Pl must be equidistant from the center point Pj. If the word NOCHECK is included as the last parameter, the radius of the arc will be defined as the distance between the points Pj and Pk, the plane of the arc will contain the point Pl, and a plane perpendicular to the plane of the arc which passes through the center and the end point of the arc will also pass through the point Pl, but no check will be made to verify that the point Pl lies on the curve thus defined.

(b) If the alphanumeric parameter al is CIRCUMFERENCE, the points P1 and P3 will be assumed to define the end points of the curve and the point P2 will be assumed to be some intermediate point along that curve.

(c) The three points Pj, Pk, and Pl must not be collinear.

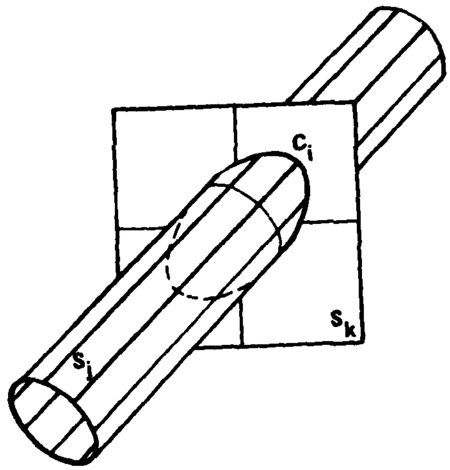
<<<:>>>

4. Curve Defined by Two Intersecting Surfaces--INTERSECTION

Function: To establish a curve variable defined to be the intersection of two surfaces.

Format: Ci, INTERSECTION, Sj, Sk [,k1 [,k2]]

Diagram:



Notes: (a) The curve defined by the intersection of surfaces Sj and Sk must be continuous (in particular, no multiple curves are permitted) and uniquely defined (does not cross itself). If multiple intersections are present, the individual curves can usually be obtained by subdivision of one or both surfaces. For additional information see the section: FINDING INTERSECTIONS.

(b) The optional integer parameters k1 and k2 define the number of points to be evaluated along the curve of intersection and the number of B-spline functions to be used to fit those points, respectively. If either parameter is omitted, the default values will be taken from the parameters set by the current FIT command. For further information see the section: DATA FITTING AND THE USE OF DIGITIZED DATA.

<<<:>>>

5. Curve Defined from a File of Digitized Data--CURVE

Function: To establish a curve variable that is defined to interpolate, fit, or approximate a set of digitized data points.

Format: Ci, CURVE, k1, k2, k3

Notes: (a) The integer constant k1 specifies the record on the digitized data input file (file name SPDATA) from which the coordinates of the

first point are to be obtained. If k1 = 0, the coordinates of all points will be obtained from subsequent records on the primary input file (e.g., keyboard input).

(b) The integer constant k2 specifies the number of points to be read from the file and used to define the curve. For format information, see the section: DATA FITTING AND THE USE OF DIGITIZED DATA.

(c) The integer constant k3 specifies the number of B-spline functions to be used to fit the set of digitized points. For further information see the section: DATA FITTING AND THE USE OF DIGITIZED DATA.

<<<:>>>

6. Curve to Fit or Interpolate a Sequence of GPRIME Points--CURVE

Function: To establish a curve variable defined to fit or interpolate a sequence of GPRIME points.

Format: Ci, CURVE, k1, <list of point variable names>

Notes: (a) The integer parameter k1 specifies the number of B-spline functions to be used to fit the data points. For further information see the section: DATA FITTING AND THE USE OF DIGITIZED DATA.

(b) <list of point variable names> includes individual point names and ranges of point names specified by inserting the alphanumeric parameter "THRU" between the beginning and ending point names of the range (e.g., P101, THRU, P107). Points not defined but included in the "THRU" range will be ignored.

(c) The curve defined will begin at (or in the vicinity of) the first point listed and will pass through (or near) the remaining points in the order listed.

<<<:>>>

7. Polygon of Points Evaluated along a Curve--EVALUATE

Function: To establish a curve variable defined to be a polygon of points evaluated at a number of stations along a curve.

Format: Ci, EVALUATE, Cj, k1

Notes: (a) This form of curve definition is primarily designed to be followed by a companion FIT definition. This EVALUATE-FIT pair can be used to obtain curves with different fitting characteristics at some time after the original definition.

(b) The integer parameter k1 is used to specify the number of points to be evaluated along the curve Cj.

<<<:>>>

8. Curve to FIT or Interpolate a Polygon of Data Points--FIT

Function: To establish a curve variable defined to fit or interpolate the data points in a polygon curve.

Format: Ci, FIT, Cj, k1

Notes: (a) This form of curve definition is used primarily to define curves that fit digitized data polygons (read with a number-of-functions parameter of zero) and to fit polygon curves produced by the EVALUATE definition statement.

(b) The parameter k1 is used to specify the number of B-spline functions to be used to fit the data in the polygon curve Cj. For further information see the section: DATA FITTING AND THE USE OF DIGITIZED DATA.

<<<:>>>

9. Parametric Definition of a Segment of a Curve--CURVE

Function: To establish a curve variable as a segment of a curve defined by the parameter values of the beginning and ending points on a reference curve.

Format: Ci, CURVE, Cj, r1, r2

Notes: (a) The real constants r1 and r2 are the parameter values of the beginning and ending points, respectively, of the subcurve along the reference curve Cj. There are no restrictions on the parameter values r1 and r2.

(b) Points along the reference curve Cj may be selected by specifying parameter values in the range zero to one. Values outside that range will yield linearly extrapolated points of the ends of the reference curve.

(c) The first parameter selected defines the beginning of the subcurve, regardless of the direction in which the reference curve was defined.

<<<:>>>

10. Interactive Definition of a Segment of a Curve--CURVE

Function: To establish a curve variable defined as a segment of a curve selected interactively from a reference curve.

Format: Ci, CURVE, Cj, TC

Notes: (a) The beginning and ending points of a subcurve are defined by two interactive graphic inputs, identifying first the beginning point and then the ending point along the reference curve Cj. The first point selected defines the beginning of the subcurve, regardless of the direction in which the reference curve was defined.

(b) After the curve has been defined, the definition will be changed to parametric form (Ci, CURVE, Cj, r1, r2) for future reference.

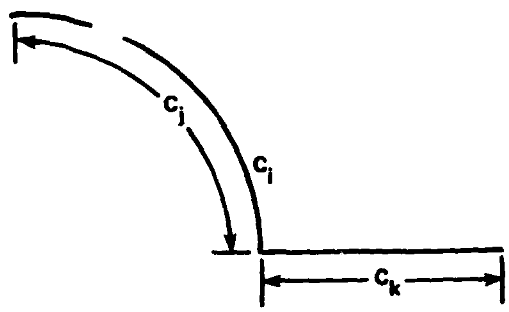
<<<:>>>

11. Curve Defined by Joining Two Curves--MERGE

Function: To establish a curve variable defined as the union of two parent curves.

Format: Ci, MERGE, Cj, Ck

Diagram:



Notes: (a) The two parent curves Cj and Ck must have at least one common end point (i.e., the point must be coincident within a tolerance established by the TOLERANCE command).

(b) Neither of the two parent curves Cj and Ck may be closed curves.

(c) As the MERGED curve Ci is traversed, the path (locus) produced will be identical to the union of the paths (loci) produced by traversing each

of the parent curves. The parameterization of the new curve will necessarily be different.

(d) Storage allocations in the GPRIME processor limit the total number of curves which can be joined using this definition. (The CDC version of GPRIME limits the user to 32 points for each curve definition. Under these conditions, eleven straight line segments could be merged; however, two complex curves may well exceed the limit.)

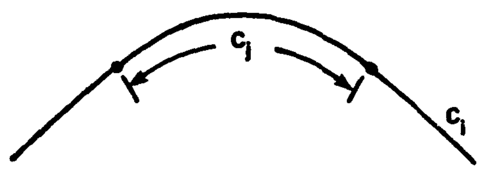
<<<:>>>

12. Curve Extended by Extrapolation--EXTEND

Function: To establish a curve variable defined as a linear extrapolation beyond the ends of a parent curve.

Format: Ci, EXTEND, Cj, r1

Diagram:



Notes: (a) The real parameter r1 defines the magnitude of the expansion of the curve (the fractional increase of the parent curve that is to be added at each end).

(b) As the extended curve Ci is traversed, the path (locus) produced will include the same path (locus) produced by traversing the parent curve Cj. The parameterization of the new curve will necessarily be different.

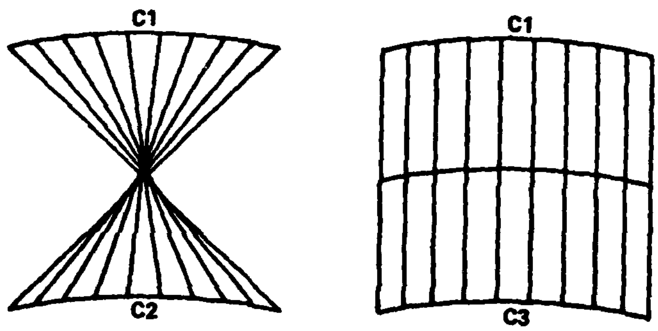
<<<:>>>

13. Curve Defined in Reverse Parametric Order--OPPOSITE

Function: To establish a curve variable defined to have a parameter direction opposite to that of a parent curve.

Format: Ci, OPPOSITE, Cj

Diagram: Ruled surfaces with C3 defined as OPPOSITE relative to C2.



Notes: (a) As the transposed curve Ci is traversed, the path (locus) produced will be identical to the path (locus) produced by traversing the parent curve. The parameterization is changed so that the point on the parent curve defined by a parameter value of s will be obtained by evaluating the transformed curve at a parameter value of 1-s.

(b) The OPPOSITE definition is used primarily to obtain sets of compatible curves for certain forms of surface definition (e.g., ruled surfaces).

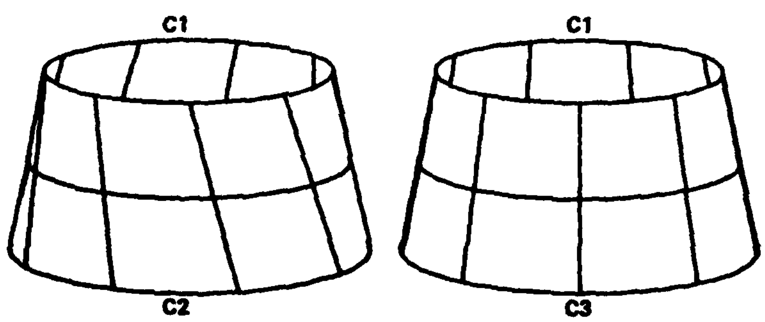
<<<:>>>

14. Closed Curves Defined by Parametric Translation--OFFSET

Function: To establish a curve variable defined to be a parametric translation of a parent curve.

Format: Ci, OFFSET, Cj, ri

Diagram: Ruled surfaces with C3 defined as OFFSET relative to C2.



Notes: (a) The real parameter r1 is the offset value to be applied to the parameter of the parent curve Cj to obtain the definition of the translated curve. The parameterization is changed so that points on the parent curve defined by a value of s will be obtained by evaluating the translated curve at a parameter value of s-r1.

(b) The OFFSET definition is used primarily to obtain sets of compatible curves for certain forms of surface definitions (e.g., ruled surfaces).

(c) The translated curve definition is obtained by first evaluating a polygon of points on the parent curve and then by fitting those points (the number of points and the number of fitting functions are governed by the FIT command). Therefore, the locus of the translated curve will not necessarily be identical to the locus of the parent curve.

<<<:>>>

15. Closed Curves Defined by Interactive Translation--OFFSET

Function: To establish a curve variable defined to have a specified parametric translation relative to a parent curve.

Format: Ci, OFFSET, Cj, TC

Notes: (a) A curve having a new pseudo beginning and ending point will be defined by interactively selecting that point on the parent curve. The parameterization of the new curve will be translated accordingly.

(b) The OFFSET definition statement is used primarily to obtain sets of compatible curves f certain forms of surface definition (i.e., ruled surfaces).

(c) The translated curve definition is obtained by first evaluating a polygon of points on the parent curve and then by fitting those points (the number of points and the number of fitting functions are governed by the FIT command). Therefore, the locus of the translated curve will not necessarily be identical to the locus of the parent curve.

(d) After the curve has been defined, the definition will be changed to the parametric form (Ci, OFFSET, Cj, r1).

SURFACE DEFINITION

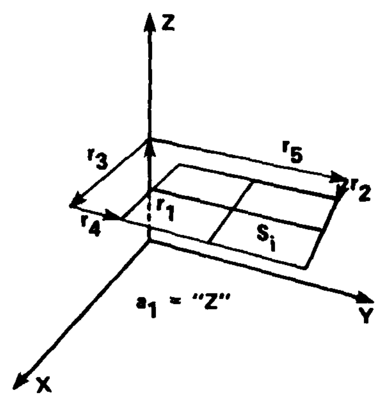
Common classical surfaces can be defined in more-or-less conventional mathematical terms with the GPRIME language. There are also several methods for defining more general surfaces, including surfaces which pass through a number of curves and surfaces defined by digitized data points. All GPRIME surfaces are considered to have finite extent; however, bilinearly extrapolated values can be obtained outside the region of original definition, when necessary. Internally, surface representations are stored as topologically rectangular meshes of points (this is also the form required for defining surfaces via externally digitized data). There are no GPRIME commands which automatically define surfaces.

1. Plane Perpendicular to a Coordinate Axis--PLANE

Function: To establish a surface variable defined as a plane perpendicular to a coordinate axis in the base rectangular coordinate system.

Format: Si, PLANE, r1, r2, r3, r4, r5, a1

Diagram:



Notes: (a) The real parameter r1 defines the value of the constant coordinate being defined.

(b) The constant coordinate direction is specified with the alphanumeric parameter a1 (either "X", "Y", or "Z").

(c) The real parameters r2 through r5 define the extent of the plane being defined. The interpretation of these parameters is governed by the

alphanumeric parameter a1 as indicated below:

a1 Value	r2 & r3 Define	r4 & r5 Define
X	y-limits	z-limits
Y	x-limits	z-limits
Z	x-limits	y-limits

(d) There are no restrictions on any of the real parameter values.

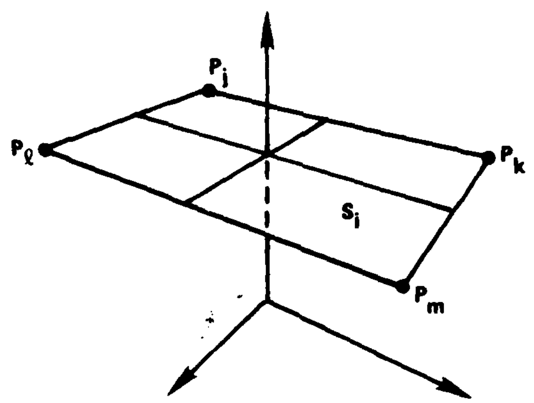
<<<:>>>

2. Plane or Warped Surface Defined by Four Points--SURFACE

Function: To establish a surface variable defined as a planar or warped quadrilateral patch, with four specified corner points.

Format: Si, SURFACE, Pj, Pk, Pl, Pm

Diagram:



Note: The lines Pj-Pk and Pl-Pm define opposite (non-adjacent) edges of the quadrilateral and the lines Pj-Pl and Pk-Pm also define opposite edges, as shown in the diagram.

<<<:>>>

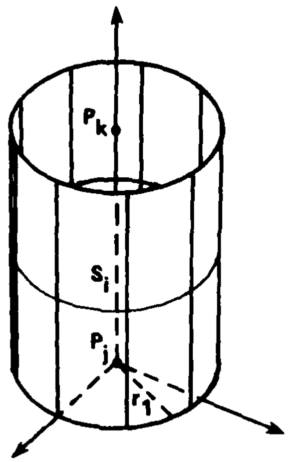
3. Right Circular Cylinder(1)--CYLINDER

Function: To establish a surface variable defined as a right circular

cylinder, given the end points of the cylindrical axis and the radius of the cylinder.

Format: Si, CYLINDER, Pj, Pk, r1

Diagram:



Note: Point Pj defines one end of the cylindrical axis and point Pk defines the opposite end. The real parameter r1 defines the radius of the cylinder.

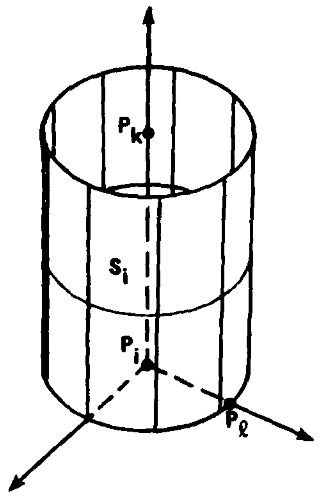
<<<:>>>

4. Right Circular Cylinder(2)--CYLINDER

Function: To establish a surface variable defined as a right circular cylinder, given the end points of the cylindrical axis and one point on the circumference of the cylinder, at the base.

Format: Si, CYLINDER, Pj, Pk, Pl

Diagram:



Notes: (a) Point Pj defines the base end of the cylindrical axis, point Pk defines the opposite end of the cylindrical axis, and point Pl is the point on the cylinder at the base.

(b) If the point Pl does not lie in the plane containing Pj, which is perpendicular to the cylindrical axis, the surface defined will be a skewed (i.e., elliptical) cylinder.

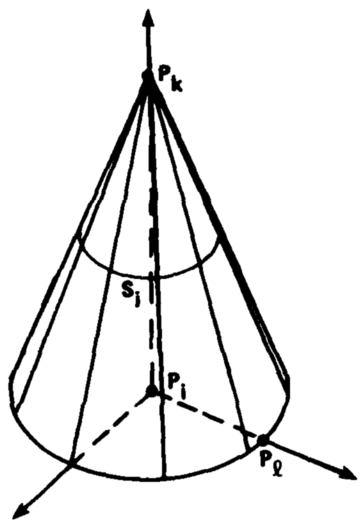
<<<:>>>

5. Circular Cone--CONE

Function: To establish a surface variable defined as a circular cone, given a point at the center of the base of the cone, a point at the apex, and a point on the cone at the base.

Format: Si, CONE, Pj, Pk, Pl

Diagram:



Notes: (a) Point Pj defines the base end of the axis of the cone, point Pk defines the apex, and Pl is a point on the cone at the base.

(b) If the point Pl does not lie in the plane containing Pj, which is perpendicular to the axis of the cone, the surface defined will be a skewed (i.e., elliptical) cone.

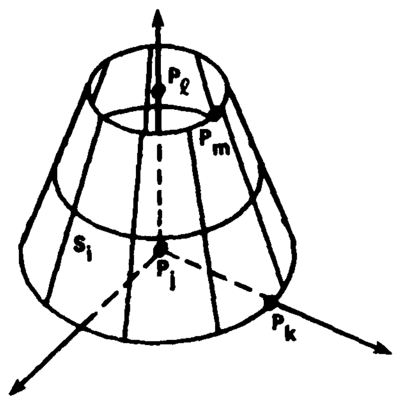
<<<:>>>

6. Frustum of a Circular Cone--FRUSTUM

Function: To establish a surface variable defined as the frustum of a circular cone, given two points on the axis of the cone and two points on the frustum.

Format: Si, FRUSTUM, Pj, Pk, Pl, Pm

Diagram:



Notes: (a) Point Pj defines the base end of the axis of the frustum and point Pl defines the opposite end of the axis. Pk is a point on the frustum at the base end and Pm is a point on the frustum at the opposite end.

(b) For proper definition of a frustum of a circular cone: (1) the points Pj through Pm must lie in a plane, (2) the point Pk must lie in a plane containing the point Pj, which is perpendicular to the axis, and (3) point Pm must lie in the plane containing the point Pl, which is perpendicular to the axis. Otherwise, a skewed surface will be defined.

<<<:>>>

7. Sphere(1)--SPHERE

Function: To establish a surface variable defined as a sphere, given its center point and radius.

Format: Si, SPHERE, Pj, r1

Notes: (a) Point Pj defines the center point of the sphere and the real parameter r1 defines the radius of the sphere.

(b) The real parameter r1 must be a positive number.

<<<:>>>

8. Sphere(2)--SPHERE

Function: To establish a surface variable defined as a sphere, given its center point and a point on the surface of the sphere.

Format: Si, SPHERE, Pj, Pk

Notes: (a) Point Pj defines the center of the sphere and the point Pk is a point on the surface of the sphere.

(b) The points Pj and Pk must not coincide.

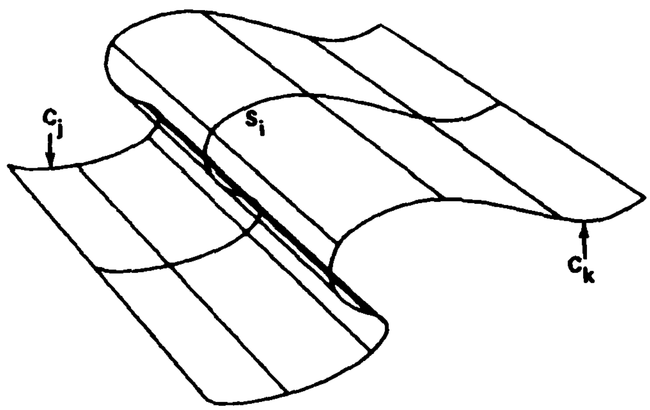
<<<:>>>

9. Ruled Surface between Two Curves--SURFACE

Function: To establish a surface variable defined as a surface of straight line generators connecting corresponding points on a pair of three-dimensional arcs.

Format: Si, SURFACE, Cj, Ck

Diagram:



Notes: (a) The curves Cj and Ck are the three-dimensional defining arcs.

(b) The GPRIME processor generates ruled surfaces by using straight-line generators to connect those points on the defining arcs which correspond parametrically. The parametric direction (and the pseudo beginning and ending points of closed curves) must be considered when defining ruled surfaces. The straight line connecting the beginning points of the two curves will always form an edge of the surface, as will the line connecting the ending points of the curves. The user may effectively define ruled surfaces which have straight line generators between the beginning of one arc and the end of the

second by first obtaining a transformed definition using the OPPOSITE definition statement. For closed curves, the pseudo beginning and ending points may be relocated using an OFFSET definition statement. For further information see the section: TRANSFORMATION OF GEOMETRIC DATA.

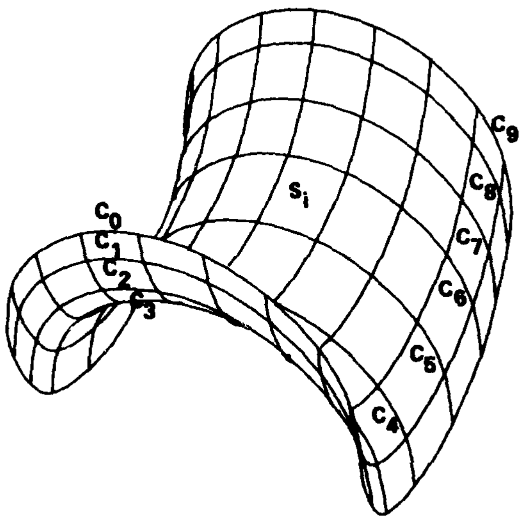
<<<:>>>

10. Surface to Fit or Interpolate a Collection of Curves--SURFACE

Function: To establish a curve variable defined to fit or pass through a collection of curves.

Format: Si, SURFACE, k1, k2, <list of curve variables>

Diagram:



Notes: (a) The integer parameter k1 indicates the number of B-spline functions to be used to fit the curve data in the direction of each curve. The integer parameter k2 indicates the number of B-spline functions to be used to fit the curve data in the direction opposite that of each curve. For further information see the section: DATA FITTING AND THE USE OF DIGITIZED DATA.

(b) The <list of curve variables> includes individual curve names and ranges of curve names specified by inserting the alphanumeric parameter "THRU" between beginning and ending curve names in the range (e.g., C10, THRU, C17). Curves not defined but included in the range of names will be ignored.

(c) The surface defined will begin at (or in the vicinity of) the

first curve listed and will pass through (or near) the remaining curves in the order listed.

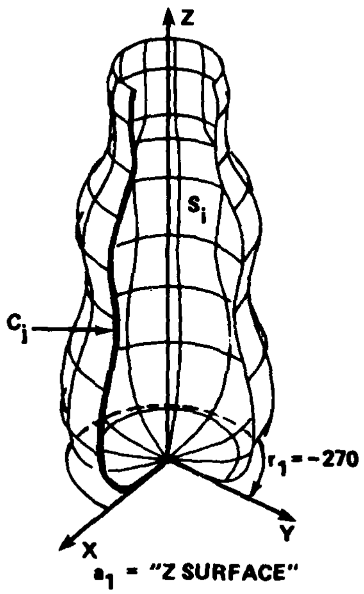
<<<:>>>

11. Surface of Rotation--XSURFACE, YSURFACE, ZSURFACE

Function : To establish a surface variable defined as the rotation of a generating curve about a coordinate axis.

Format: Si, a1, Cj [,r1]

Diagram:



Notes: (a) The command name a1 is either XSURFACE, for a surface defined by rotating the curve Cj about the x-axis; YSURFACE, for a surface defined by rotating Cj about the y-axis; or ZSURFACE, for a surface defined by rotating Cj about the z-axis.

(b) The optional real parameter r1 defines the angle (in degrees, right-hand rule) through which the generating curve Cj will be rotated. If r1 is not specified, 360 degree rotation is assumed.

<<<:>>>

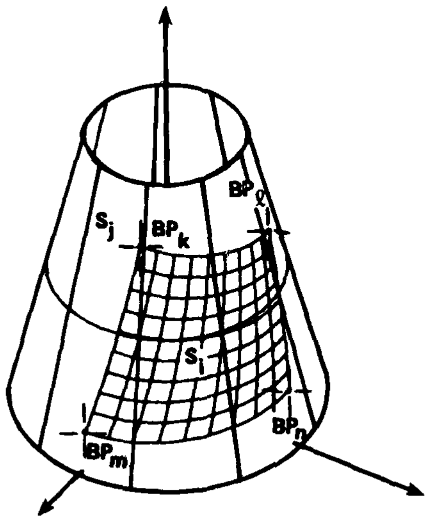
12. Simple Parametric Subsurface--SURFACE

Function: To establish a surface variable defined as the subregion

of a GPRIME surface, bounded by parametric straight lines connecting four corner points.

Format: Si, SURFACE, Sj, BPk, BPl, BPm, BPn

Diagram:



Notes: (a) The B-spline parameter defining the corners of the subregion (BPk, BPl, BPm, and BPn) may be defined "in-line" by enclosing the pair of real parameter values in parentheses (e.g., (.5, .7)).

(b) There is no restriction on the range of parameter values, but if any of the values are outside the range zero to one, bilinearly extrapolated surface values will be used.

(c) The parametric lines BPk-BPl and BPm-BPn will define opposite edges of the subsurface as will the lines BPk-BPm and BPl-BPn.

(d) Parametric straight lines (linear functions in the parameter space) follow the general trend of the reference surface between the two end points. Only for very simple surfaces will these be straight lines in the object space.

<<<:>>>

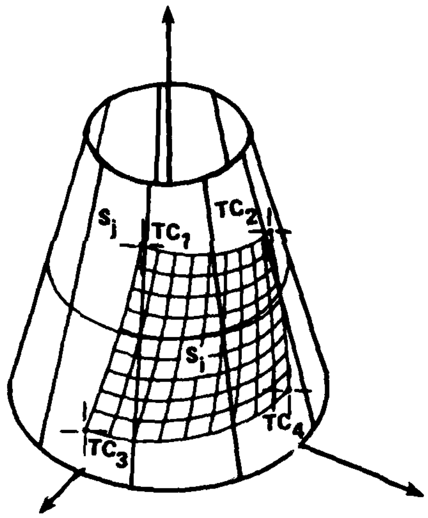
13. Interactively Defined Parametric Subsurface--SURFACE

Function: To establish a surface variable which is defined by interactively selecting four corner points of a subregion on a reference

surface. The subregion will be bounded by four parametrically straight lines on the reference surface.

Format: Si, SURFACE, Sj, TC

Diagram:



Notes: (a) Four corner points of the subregion are defined on the reference surface Sj using interactive graphic input from the user's terminal.

(b) Parametric straight lines connecting the first and second points and the third and fourth points will define opposite edges of the subsurface, as will the lines connecting the first and third points and the second and fourth points.

(c) After the surface has been defined, the definition will be changed to parametric form (Si, SURFACE, Sj, BPk, BPl, BPm, BPn).

(d) Parametric straight lines (linear functions in the parameter space) follow the general trend of the reference surface between the two end points. Only for very simple surfaces will these be straight lines in the object space.

<<<:>>>

14. Surface Defined from a File of Digitized Data--SURFACE

Function: To establish a surface variable defined to interpolate, fit, or approximate a set of digitized data points.

Format: Si, SURFACE, k1, k2, k3, k4, k5

Notes: (a) The integer constant k1 specifies the record on the digitized data input file (file name SPDATA) from which the coordinates of the first point are to be obtained. If k1=0, the coordinates of all points will be obtained from subsequent records from the primary input file (e.g., keyboard input).

(b) The integer constants k2 and k3 define the size of the rectangular array of data points to be read. k3 curves (polygons), of k2 points each, will be used to define the surface.

(c) The integer constant k4 specifies the number of B-spline functions to be used to fit the set of data in the direction along the data curves, and the constant k5 specifies the number of B-spline functions to be used to fit the data in the cross-curve direction. For further information see the section: DATA FITTING AND THE USE OF DIGITIZED DATA.

<<<:>>>

15. Mesh of Points Evaluated on a Surface--EVALUATE

Function: To establish a surface variable as a rectangular mesh of points evaluated at a number of stations on a surface.

Format: Si, EVALUATE, Sj, k1, k2

Notes: (a) This form of definition is designed primarily to be followed by a companion FIT definition. This EVALUATE-FIT pair can be used to obtain surfaces with different fitting characteristics at some point after the original definition.

(b) The integer parameter k1 is used to specify the number of points to be evaluated along one curve running across the surface to form one polygon of the mesh. The integer parameter k2 is used to specify the number of such polygons to be used for the mesh. This definition results in a rectangular mesh consisting of k2 polygons, each polygon being composed of k1 points.

(c) The spacing of the points to be evaluated is governed by uniform increments in the parameter space.

<<<:>>>

16. Surface to Fit or Interpolate a Mesh of Points--FIT

Function: To establish a surface variable defined to fit or interpolate the data points in a topologically rectangular mesh of points.

Format: Si, FIT, Sj, k1, k2

Notes: (a) This form of a surface definition is used primarily to define surfaces from meshes of digitized data points from rectangular mesh surfaces produced by the EVALUATE definition statement.

(b) The integer parameter k1 is used to specify the number of B-spline functions to be used to fit the data in the curve-wise direction of the mesh. The integer parameter k2 is used to specify the number of B-spline functions to be used to fit the data in the cross-curve direction of the mesh. For further information see the section: DATA FITTING AND THE USE OF DIGITIZED DATA.

<<<:>>>

17. Parametrically Transformed Surface--OPPOSITE

Function: To establish a surface variable defined as a parametric transformation of a parent surface.

Format: Si, OPPOSITE, Sj, a1

Notes: (a) The type of transformation to be applied to the parent surface definition Si is governed by the alphanumeric character a1. If:

a1 = S-DIRECTION, transform the definition such that the new curve-wise parameter s' is related to the parent surface parameter s by $s' = (1-s)$.

a1 = T-DIRECTION, transform the definition such that the new cross-curve parameter t' is related to the parent surface parameter t by $t' = (1-t)$.

a1 = BOTH, apply both of the above transformations.

a1 = TRANSPOSE, transpose the definition such that the values of the curve-wise parameter and the cross-curve parameter are interchanged.

(b) Only the parameterization of the definitions will change under the above transformations. The point obtained by evaluating the transformed surface Si with a given parameter pair (s,t) will be identical to the point obtained by evaluating the parent surface Sj with a transformed parameter pair (s',t').

(c) These transformations are frequently used to alter the order of finite element generation in programs like the GGEN data generator.

GROUP DEFINITION

GPRIME GROUPS provide a means of collecting geometric elements for convenient manipulation. All elements within a group must be defined. There is only one type of group definition statement, but the GENERATE, POINTS command and the GGEN data generator also define groups. The identity transformation is the only geometric transformation that can currently operate on group variables, and CHANGE and DELETE are the only GPRIME commands which currently operate on groups. Programs and commands which use GPRIME groups may impose further restrictions on the format of those groups (e.g., the GGEN data generator permits only one reference to a continuation group and that must be the last element in the group).

Group of GPRIME Elements--GROUP

Function: To establish a group variable defined as a list of GPRIME geometric elements.

Format: Gi, GROUP, <list of element names>

Notes: (a) The number of elements in the <list of elements> must not exceed 25. Group names may be included in the list to define effectively longer lists.

(b) Ranges of element names may be included by bracketing the parameter "THRU" between the beginning and ending names of the elements in the range. Variable names included within a range but not defined will be ignored.

TRANSFORMATION OF GEOMETRIC DATA

The GPRIME language has two definition forms which permit the user to define elements by applying general geometric transformations to existing elements. The translate-and-scale definition form (TRSCALE) is used in the same way as all other GPRIME definitions; however, GPRIME'S ORIENTATION transformation, which can be used to define rotated and translated geometry, requires a modification to the usual definition procedure. First, orientation transformations are established as GPRIME variables and as such can be used to define many geometric elements and other orientation transformations as well. Second, the orientation transformation can be applied to most geometric definitions by including the name of the desired orientation transformation as the last parameter of the definition (e.g., C1, LIN, P1, P2, OT3).

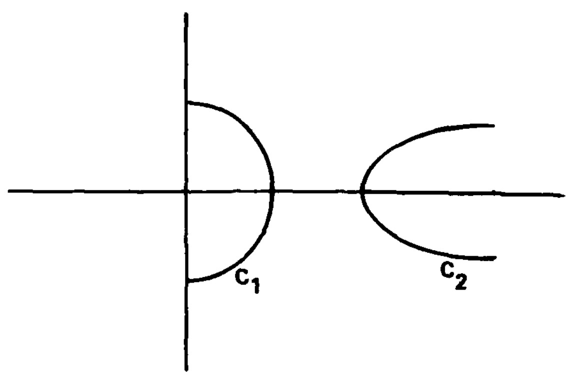
A third form of data transformation available within GPRIME is simple replication, which is just an identity operator. The identity definition is a convenient device for applying orientation transformations to existing geometric elements.

1. Translation and Scaling of Geometric Elements--TRSCALE

Function: To establish a GPRIME variable defined as a scaled and translated version of an existing geometric element.

Format: <a1><k1>,TRSCALE,<a1><k2>,r1,r2,r3 [,Pi]

Diagram: C2, TRSCALE, C1,-1,.5,1,(10,0,0)



Notes: (a) The real parameters r1, r2, and r3 are multipliers to be applied to the x-, y-, and z-coordinates, respectively, of the parent element (the <a1><k2> parameter) of the definition.

(b) The optional point variable Pi is interpreted as a translation vector to be applied to the coordinates of the parent element. This parameter can be omitted if no translation is desired.

(c) The following types of geometric elements may be transformed with this form of definition: POINTS, CURVES, SURFACES.

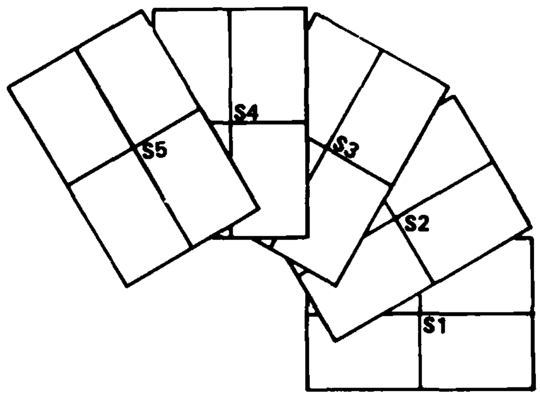
<<<:>>>

2. General Rotation and Translation Definition--ORIENTATION

Function: To establish a transformation variable defined by rotation angles and a translation vector.

Format: OTi, ORIENTATION, r1, r2, r3 [, Pj]

Diagram: OT1,ORIENTATION,0,0,30,(0,0,2); Si+1,Si, OT1



Notes: (a) The real parameters r1, r2, and r3 define angles of rotation (in degrees, right-hand rule) about the x-, y-, and z-axes, respectively. The x-rotation is applied first, followed by the y-rotation and then the z-rotation.

(b) The optional point Pj is interpreted as a translation vector. This parameter can be omitted if no translation is desired.

(c) The following types of geometric elements may be transformed using an orientation transformation: POINTS, CURVES, SURFACES, and ORIENTATION TRANSFORMATIONS.

(d) Orientation transformations are applied by including the name of the orientation transformation as the last parameter of any appropriate definition statement.

<<<:>>>

3. Replication of Geometric Elements

Function: To establish a GPRIME variable defined to be identical to a parent element.

Format: <a1><k1>, <a1><k2> [,OTi]

Notes: (a) <a1><k1> and <a1><k2> are the newly defined and parent variable names, respectively.

(b) Any type of geometric element may be replicated using this form of definition.

(c) The orientation transformation parameter OTi is optional for all GPRIME definition statements that define geometric elements in three-dimensional space. Such transformations are frequently used with replication definitions to maintain both the basic and transformed elements.

PARAMETER DEFINITION

This implementation of the GPRIME language is based mathematically on parametric B-spline functions. It is sometimes useful to define elements in terms of a B-spline definition and associated parameters (especially for archiving purposes), and a special parameter variable type has been established for this purpose. These variables may be defined either explicitly, in a definition statement, or implicitly, within other definition statements. Parameter variables are two-dimensional vectors used in conjunction with curve and surface definitions. For curve definitions, the second component of the vector is ignored. There are no GPRIME <u>commands</u> which automatically define parameter variables.

1. *Basic B-Spline Parameter Definition*--BPOINT

Function: To establish a parameter variable and assign a value to that variable.

Format: BPi, BPOINT, r1 [,r2]

Note: The real constants r1 and r2 define the components of the parameter variable. If the optional parameter r2 is omitted, a default value of zero is assumed.

<<<:>>>

2. In-Line Parameter Definition--(s,t)

Function: To establish a parameter variable from within a parent definition statement and to assign a value to that variable.

Format: ..., (r1 [,r2]), ...

Notes: (a) The real constants r1 and r2 define the components of the parameter variable. If the optional parameter r2 is omitted, a default value of zero is assumed.

(b) The GPRIME processor will assign a unique variable name to each parameter variable defined "in-line".

B-SPLINE FUNCTIONS

Ideally, the user of a geometric language should not have to be concerned with the techniques used to implement that language on a computer. The GPRIME user will find, however, that an awareness of the underlying mathematics will occasionally be required to produce certain constructions and that such an awareness can often lead to more efficient use of GPRIME. It is particularly important that the user be aware that all curves and surfaces defined within GPRIME will be stored internally as parametric cubic B-spline approximations.

The B-spline functions used for curve and surface fitting by the GPRIME processor are known as cardinal cubic B-splines. Although this subset of B-spline functions may not always satisfy the automobile body designer, its capability for modeling most manufactured items and for creating finite element models of structures has been clearly demonstrated. A more general B-spline basis could have been used in our implementation of the GPRIME processor, but the list of options from which the program user must make appropriate selections very quickly becomes long and unwieldy. Even under the current implementation, the program user must still consider a few options related to the underlying mathematical basis; however, most of these considerations arise in connection with data fitting and the use of digitized data.

As the GPRIME project has progressed, interest in and literature on B-splines has greatly increased. Most of the GPRIME mathematics subroutines have grown from the ideas presented in a thesis by Richard Riesenfeld.[4] Many of the B-spline techniques we use have been developed specifically for GPRIME[5] (the intersection and fitting capabilities, for example). For additional information the recent book A Practical Guide to Splines by Carl deBoor,[6] is a good general reference on the subject.

The question, "What are B-spline functions?" is difficult to answer briefly and effectively. Mathematically, the form is much like that of other approximation techniques; we choose to approximate some function by a linear combination of certain basis functions. Thus, the point Qm on a parametric B-spline curve induced by a polygon of m points P is given by:

$$Qm\,[\,\underline{P}\,;s] = \sum_{i=1}^{m} Ni(s)\,Pi\ ;\qquad 0 \le s \le 1$$

For our GPRIME implementation, the points Qm [$\underline{P}$,s] and Pi are understood to be three-dimensional vectors. Ni(s) is the i^{th} normalized cubic B-spline basis function evaluated at the point s. A set of B-spline basis functions for m=9 is shown in Figure 7, where a typical basis function, centered about the point xi, is drawn as a solid curve.

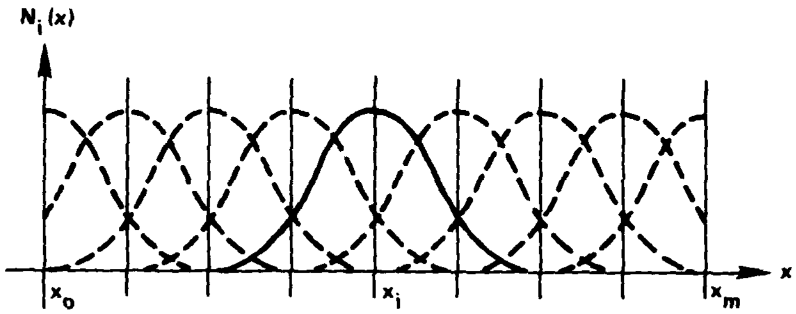


Figure 7 - A B-Spline Basis

Each cubic B-spline basis function is a piecewise cubic polynomial which has second-derivative continuity everywhere and third-derivative continuity everywhere except at the "knot points" (the points at which the cubic polynomials are pieced together). Any linear combination of these basis functions (which include the B-spline functions, of course) will have the same order of continuity. GPRIME surfaces are Cartesian product generalizations of these B-spline curves and are functions of two parameters s and t in the form:

$$Qmn \left[[P] ; s,t \right] = \sum_{i=1}^{m} \sum_{j=1}^{n} Ni(s)\ Nj(t)\ Pij \ ; \quad 0 \leq s \leq 1,\ 0 \leq t \leq 1$$

For direct B-spline approximation, the polygon $\underline{P}$ is taken to be a set of points on the curve to be approximated. For least-squares fitted curves and for interpolating curves, $\underline{P}$ represents a set of coefficients to be determined from a polygon of data points $\underline{Q}$.

To relate B-splines to the more familiar splines used for the past thirty years,[7] note that a B-spline curve used by GPRIME to interpolate a set of m

points is the same as the cubic cardinal spline with m knots which passes through those points.

All GPRIME curves are defined as parametric functions of finite extent. The curve representations are normalized so that a parameter value of zero is associated with the beginning of the curve and a value of one is associated with the end. Similarly, surfaces are defined as functions of two parameters, which also have finite extent. Parameter values of zero and one are associated with the "edges" of the surface. A few GPRIME definition statements may include the optional use of parameter values. The GPRIME user will not ordinarily set those parameters, but programs that record interactive transactions via GPRIME statements may choose to use them to maintain compatibility with the B-spline subroutine calls being used.[2] Whenever parameter values are specified outside the range zero to one, the processor will yield points obtained by linear (bilinear) extrapolation beyond the extent of the curve (surface).

GPRIME permits the user to define curves and surfaces which close upon themselves. These *closed representations* have the same order of continuity throughout, so that there is no unique point or curve of closure. However, it is often convenient (and sometimes necessary) to know where such a closed curve or surface "begins" or "ends"; that is, to know the locations on the curve or surface which correspond to parameter values of zero and one. For closed classical shapes generated by GPRIME, the first reference point or curve encountered in the definition, excluding centers of rotation, will be defined as the beginning. Closed curves will be generated from digitized data whenever the first and last data points have identical coordinates. Closed surfaces will be generated from digitized data whenever all the data points on the first and last curves, in either direction, coincide (within some tolerance, as defined by the TOLERANCE command). These curves and surfaces are defined to begin and end at the points or curves that correspond to the first data points or data curves given.

Some closed curves, particularly curves of intersection, may be generated from a starting point that is not compatible with the intended application, or they may be generated in the reverse order. Such problems can be handled using the OFFSET and OPPOSITE definition statements.

GPRIME's internal definitions can be displayed by invoking the BSPLINE command. For variables other than curve and surface variables, the internal definitions are in a form that can be directly interpreted by the user. Curve and surface definitions are stored as arrays of points which induce the required B-spline curve or surface. In general, these points do not lie on the curve or surface defined by the variable in question, and thus are not appropriate for verifying a definition if used by themselves. Curves and surfaces can be verified by first choosing a set of reference points on the curve or surface (see POINTS.5, for example) and then displaying the internal definitions of those points.

DATA FITTING AND THE USE OF DIGITIZED DATA

Certainly, a GPRIME user will be concerned that the program will represent his geometry accurately. Default data fitting parameters have been provided which produce accurate results for most problems, but the user is also given the capability to intervene in the fitting process. A classical curve defined within GPRIME (e.g., a circle) will be approximated by evaluating that curve at a number of points and fitting those points with half that number of B-spline functions. The FIT command is used to set the number of evaluation points. A classical surface (e.g., a sphere) will be approximated by evaluating that surface to obtain points on a number of curves across the surface. The FIT command is used to set the number of points to be evaluated on each curve and to set the number of curves to be used. The surface will then be approximated by fitting the points obtained, using half that number of B-spline functions in each direction. If, for example, the FIT parameter is set to 32, 32 points will be evaluated on each curve and the internal definition will consist of 16 B-spline coefficients. Similarly, a surface will be evaluated to obtain a 32 by 32 array of points, which, after fitting, will result in a 16 by 16 array of B-spline coefficients stored as the internal representation.

The following observations give an indication of the accuracy of these least-squares fitted approximations. Evaluating a quarter circle at 16 equally spaced points and fitting those points using 8 B-spline functions will give errors of approximately 0.2 percent. Evaluating a quarter circle at 32

equally spaced points (close to the maximum number of points for CDC GPRIME) and fitting them using 16 B-spline functions will give errors of 0.02 percent. Unfortunately, users who require greater accuracy will find that they must pay close attention to each fitting operation and make adjustments as necessary. Greater accuracy can be obtained for specific points by setting the number of functions equal to the number of points evaluated (interpolation). The accuracy can always be improved by subdividing the original element into two or more subelements.

More user interaction is required to fit externally digitized data. In addition to specifying the number of points that have been supplied, the user must also specify the number of B-spline functions to be used for fitting. If the number-of-functions parameter is set equal to the number of points to be fitted, an interpolating B-spline curve will be produced which passes through each of the data points. However, the curvature of interpolating curves usually changes too abruptly for satisfactory representation of the original data. Very smooth curves can be obtained by specifying a direct B-spline approximation (number-of-functions parameter is set to zero), but this approximation will always miss the data at the peaks and valleys. Least-squares fitting is usually the most satisfactory technique for representing externally digitized data. Least-squares fitting is selected by setting the number of functions parameter to a value greater than zero and less than the number of points to be fitted.

One rule of thumb for determining the number of B-spline functions required to represent a given curve is to use the same number of functions as the number of points required to construct a polygon that will trace out the essential characteristics of the curve. Figure 8 illustrates some of the B-spline fitting concepts with a set of 21 points digitized from a section of a French curve. In Figure 8-a the points are shown along with the direct B-spline approximation curve induced by those points (non-fitted, number of functions parameter set to zero). Note that this curve would lie on the concave side of an open polygon connecting all the points in sequence. Figure 8-b shows the same points with an interpolating B-spline curve (number of functions set to 21). Figure 8-c shows the 21 points with a polygon having 12 vertices which seems to capture the character of the data (peaks, valleys,

and points of inflection). Figure 8-d shows a least-squares-fitted curve (number of functions set to 12) which appears to be a good approximation to the original curve data. This rule works very well when the data points are uniformly spaced.

The user can indicate that no fitting is to be performed when digitized data are being read from a file, and the FIT definition statement can be used for fitting the data at a later time. Curves of intersection are treated like classical curves unless the user chooses to override the default values on the INTERSECTION definition statement.

Straight line curves and the linearly varying nature of ruled surfaces can always be accurately represented using two B-spline functions. Internally generated curves and surfaces with these properties will be represented by only two B-spline functions and not by the number indicated with the FIT parameter.

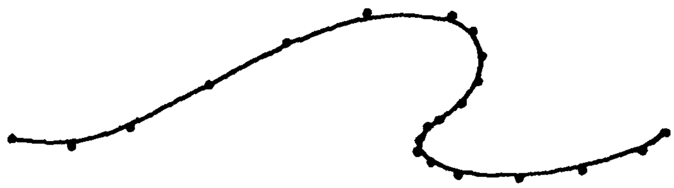


Figure 8a - B-Spline Approximating Curve

Figure 8b - Interpolating B-Spline Curve

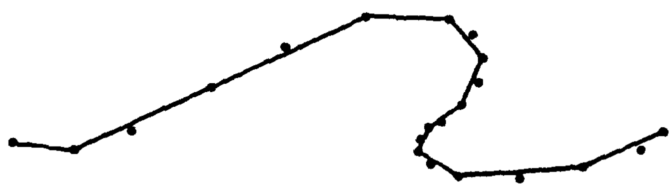


Figure 8c - "Essential Characteristics" Polygon

Figure 8d - Least-Squares-Fitted Curve

Figure 8 - Curves Approximating French Curve Data

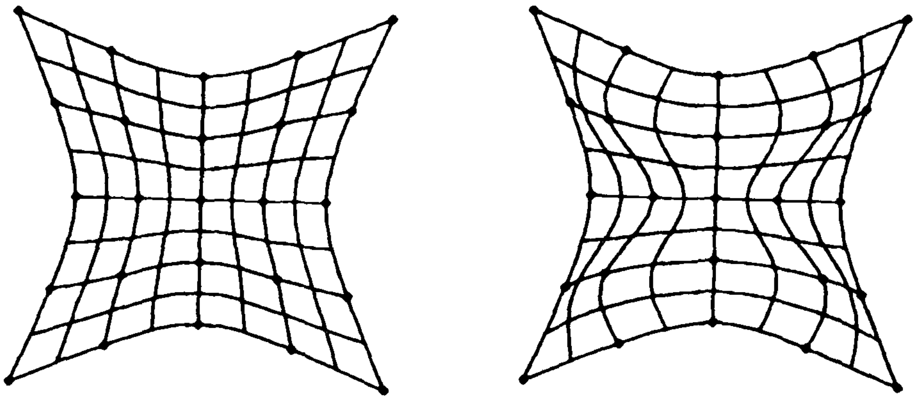
The GPRIME processor accepts externally digitized data from a special data file named SPDATA. Data on this file must be in character format (a text file). Each point defined on this file must have three coordinate-data fields separated by blanks or commas. Each data line defines one point, and any missed or blank fields will be interpreted as zero by the processor. FORTRAN-style real values will be accepted by the processor. Integer numbers are acceptable as real values on the digitized data file (as they are throughout the GPRIME language). The two following sets of data could be used to define the same three data points via the SPDATA file:

```
(Set 1)  1.0     1.0     1.0
         5.E-4   5.E-4   1.E3
         0.      0.      3.
(Set 2)  1,1,1
         .0005,.0005,1000
         0,,3
```

Note: On CDC computers, FORTRAN exponential format real numbers are not limited to ten-character fields as they are for definitions.

Unfortunately, the definition of arbitrary surfaces is something of an art. There are many valid procedures for selecting points to define a surface. Although each such procedure will define a smooth surface which is acceptable to the user, some of these procedures will yield surfaces that lead to failures in later processing. These failures usually involve some sort of intersection processing. GPRIME's intersection solvers can be made to find these troublesome intersections, but at a large penalty in processing time (as discussed later under FINDING INTERSECTIONS). However, most of these problems will vanish if the surfaces are defined using more "regular" (linear) strategies from the outset. Automatically generated finite element meshes produced by the GGEN program will follow parametric contours--further motivation to strive for regularity.

Usually, more regularity implies that the points defining the surface are arranged in a pattern that more closely resembles a square mesh than the pattern for a similar, but more nonlinear, mesh. Under these guidelines, the definition of the planar surface in Figure 9a is more regular than the definition of the equivalent surface shown in Figure 9b.



a b

Figure 9 - Non-linearity in Surface Parameterizations

Although complex geometries necessarily produce highly nonlinear surface definitions, many problems can be avoided if regular patterns are chosen when the user has an option.

This problem of regularity can be further exaggerated when surfaces are defined through a set of GPRIME curves. When each curve is defined by a different process, the user has little control over the parameterization. However, when the curves are defined by a digitizing process, the above guidelines should be applied. For example, the points and curves in Figure 10a produce the fairly regular parameterization in Figure 10b. Similar points and curves shown in Figure 10c yield the rather nonlinear equivalent surface in Figure 10d.

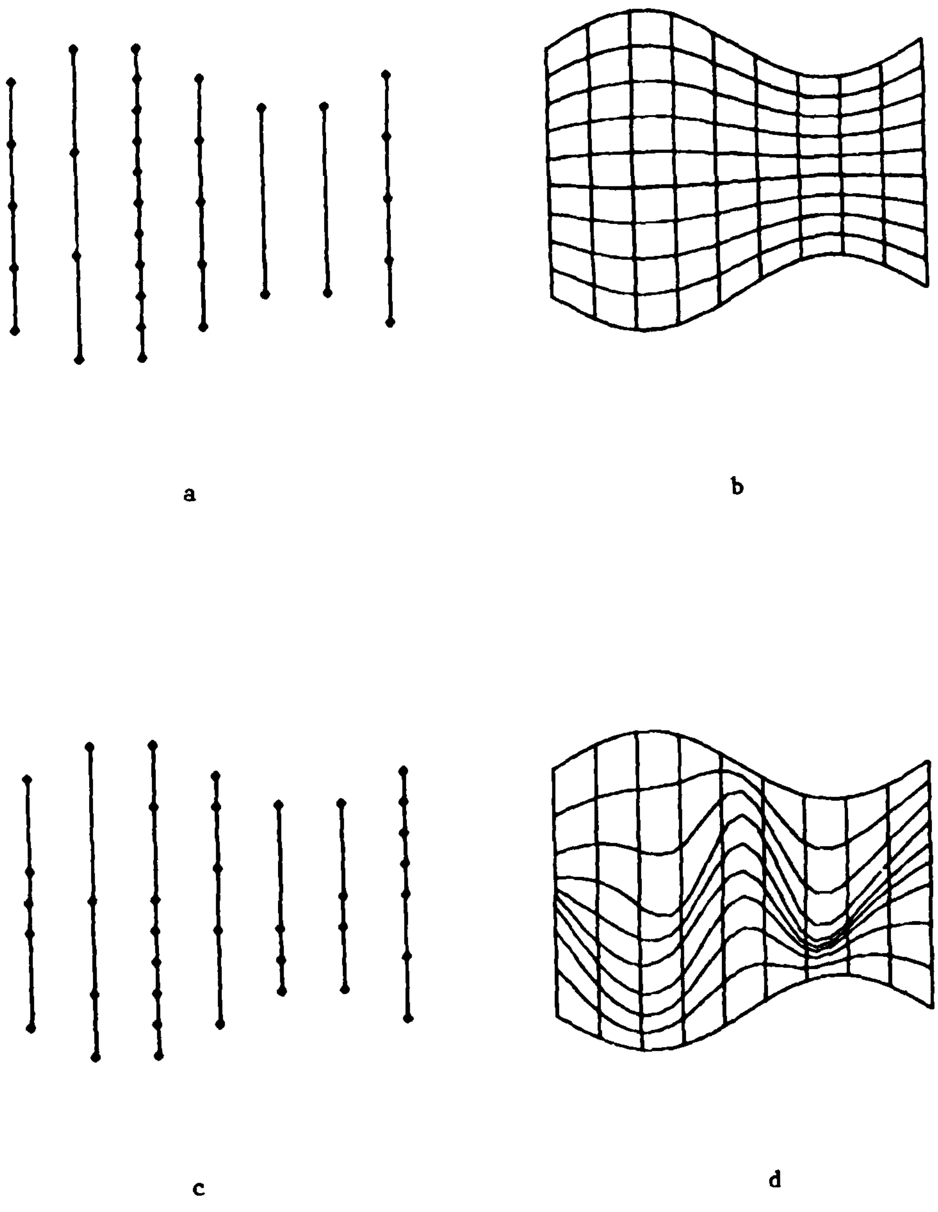


Figure 10 - Surfaces Defined Through Curves

(More consistent regularity might be achieved with the multiple-curve form of surface definition if points were chosen with equal arc-length spacing. This can now be accomplished in two steps: (1) use the "GENERATE, ARCLENGTH" command to evaluate equally spaced points and (2) define an intermediate curve using those points.)

FINDING INTERSECTIONS

All types of intersections which can be defined using the GPRIME language will be calculated using one basic iterative method. The success of this method depends on a number of factors, including estimated starting values, convergence tolerances, limits on the number of iterations, and the complexity of the curves and surfaces involved.

GPRIME's iterative solution technique is based on a non-linear minimization algorithm which uses the curve and surface parameters as the independent variables. The parametric centers of the curves and surfaces are used as initial values for the iterative solution. An intersection point is assumed whenever the set of parameter values yields points on each operand which coincide within a specified tolerance (distance .LE. 10.**-TOL, where TOL is the tolerance established by the TOLERANCE command).

GPRIME uses linear and bilinear extensions of the curves and surfaces to improve the continuity of the functions near 0.0 and 1.0 values of the parameters. This feature improves the efficiency of the iterative solution, but it does sometimes yield points which are outside the range of the basic definitions for the curve and surface operands. The GPRIME processor issues a "POINT OUT OF RANGE" message when such intersections occur.

For the interactive definition of points, the GPRIME processor also uses the iterative solver. When points are located on the extended curves or surfaces but are reasonably close to the basic element, the closest point on the basic element is assumed to be the desired point. If there are multiple points of intersection, the default initial values will usually cause the iterative solver to select the intersection point closest to the parametric centers of the operands.

Discrimination among several multiple intersections and the "POINT OUT OF RANGE" type of *false multiple intersections* can usually be accomplished by supplying a "guide point". A guide point must be clearly closer to the desired point than it is to any other intersection point. When such a guide point is given, the intersection solver first finds the point on each of the operands which is closest to the guide point and then uses those points as the initial value for the intersection solution. Unfortunately, determination of close points is often much more difficult than the intersection problem, and the user may have to subdivide the problem to obtain the desired point.

The GGEN and SOLIDGEN data generators make extensive use of GPRIME's intersection capabilities. These generators and GPRIME's POINTS command can find many points of intersection in only a few iterations, because they use the parameter values associated with the previous point as the initial value in solving for the next point. The solution technique for curves of intersection also uses previously calculated quantities as starting values for computing a sequence of points along the curve of intersection.

Curves of intersection between two surfaces are located by a technique we refer to as edge analysis. In this process, an attempt is made to intersect each edge curve of one surface with the other surface. If, after all eight edges have been examined, two distinct edge points have been located, the GPRIME processor will attempt to calculate a sequence of points on the curve of intersection which connects the two edge points. For open surfaces, there will be no attempt to determine curves of intersection unless the edge analysis yields exactly two points. For closed surfaces, one distinct edge intersection is an acceptable condition for finding a curve of intersection, if the point found lies on the curve of closure (the "seam") of a closed surface. Multiple curves of intersection, and single curves which do not meet the edge criterion, can usually be found by subdivision of the problem.

The usual reason for unexpected intersection failure in two-surface intersection problems (we expect failure if there is no intersection) is that one or more of the surface operands are highly non-linear. Most of these problems could be solved if GPRIME's solvers were more persistent. These solvers can be tuned to succeed in all but the most difficult cases, but at a very large penalty in processing time, even for simple linear problems. This

across-the-board penalty can be better understood when one considers that the surface/surface intersection procedures attempt to find many points of intersection. Most of these attempts are expected to fail, but are necessary in locating the curve of intersection. Thus, as the user increases the algorithm's persistence, the effort required to arrive at a failure conclusion is increased for all failures, hence for all intersections.

The most appropriate solution is to give the user control over the persistence of the intersection solvers. Until that capability is available, the guidelines given in the DATA FITTING AND THE USE OF DIGITIZED DATA section will minimize unexpected intersection failures.

DEFINITION INDEX

In this GEOMETRIC DEFINITION section, the GPRIME definitions have been grouped by basic geometric type and each definition is numbered within its basic group. This index references definitions by their basic group type and the definition number(s) within the group (for example, SCALAR.2 references the second scalar-type definition and POINT.3,4,6 references the third, fourth, and sixth point-type definitions). Page numbers for the basic groups are listed in the TABLE OF CONTENTS. The keywords listed in this index have been extracted from both the GPRIME definitions and the text associated with the definitions. The keywords listed in upper-case characters appear explicitly in the GPRIME definitions.

vector . . . SCALAR.2, POINT.1-8, PARAMETER.0

warped quadrilateral . . . SURFACE.2

warped surface . . . SURFACE.2

X . . . SURFACE.1

(x,y,z) . . . POINT.2

x-plane . . . SURFACE.1

XSURFACE . . . SURFACE.11

Y . . . SURFACE.1

y-plane . . . SURFACE.1

YSURFACE . . . SURFACE.11

Z . . . SURFACE.1

z-plane . . . SURFACE.1

ZSURFACE . . . SURFACE.11

CONTROL OF PROCESSING--GPRIME COMMANDS

In many ways the geometric definitions are the essence of GPRIME. Commands add a new dimension to the language by giving the user control over most aspects of the modeling process. These controls include: shorthand methods for entering a sequence of related definitions, facilities for deleting and changing definitions, communication with data generation programs, and, of course, a wide range of graphic display options. The names of the current GPRIME commands are listed in Table 1.

TABLE 1 - GPRIME COMMAND KEYWORDS

AXES	DELETE	GENERATE	OUTLINE	SAVE
BACKUP	DISPLAY	HARDCOPY	PAUSE	SCREEN
BSPLINE	DRAW	HELP	PLOT	STATUS
BUILD	DUMP	HIDE	POINTS	SURFACES
CHANGE	EDIT	IDENTIFY	PURGE	TOLERANCE
CLIP	EQUIPMENT	LABEL	REPEAT	VDT
COMMENT	EXECUTE	LIST	RESTORE	VIEW
CURVES	FIT	MATH	ROTATE	VNT
				WIPE

To begin execution of the GPRIME processor a sequence of system directives is required. These directives depend on the host computer system and the way GPRIME has been implemented at a particular site. On the CDC NOS/BE systems at DTNSRDC the following two directives are required:

ATTACH,GPRIME,ID=CAMK.

GPRIME,problemid,userid [,NEW] [,INFILE=passiveinfile].

These directives expand into a longer sequence of system directives which handle the various housekeeping tasks required for a GPRIME session.

During an interactive GPRIME session the user may obtain information about GPRIME commands by invoking the HELP command. The "HELP, COMMAND" command will produce a listing of all possible command type keys and the "HELP, COMMAND, <command type key>" will produce a short description of the specified command.

In this section command descriptions are grouped in related categories. Variable names and parameter definitions follow the conventions established in the DEFINITION STATEMENT SYNTAX section of this report. A command index is included at the end of the section for alphabetical reference.

INITIALIZATION

The GPRIME processor can be used interactively, passively, or in a combination of interactive and passive modes. Passive operation requires that a file of commands be prepared prior to running the processor. These passive input files could be created with a text editing program or prepared as a deck of punched cards. Interactive input is entered from the terminal keyboard and other graphic input devices as the program runs. The required initialization commands define the mode of operation for the current session or run. The interactive user is required to enter initialization commands, without the prompting of the program, until the interactive mode of operation has been established.

Begin a New GPRIME Project--INITIAL

Function: To indicate that this session or run is the first in a sequence of sessions or runs.

Format: INITIAL [,a1]

(a) Required first command for all sessions and runs which are not to be restarted from a previously created data base (UMF file).

Notes: (b) The optional alphanumeric parameter a1 is used to select installation-dependent modifications to the basic GPRIME program, if they are required (not required at DTNSRDC).

<<<:>>>

Defining Graphic Data Transmission Rate--RATE

Function: To set a graphic data transmission rate parameter which is required for most Tektronix storage-tube-type terminals.

Format: RATE, k1

Notes: (a) The integer parameter k1 defines the current graphic data transmission rate in characters per second. The set of acceptable values for k1 is [30, 120, 480, 960]. The default value for k1 is 120.

(b) This command must be entered prior to the Textronix mode-setting command "INTERACTIVE, STORAGE", if it is to be in effect during the current session.

(c) The graphic parameters set by this command are saved in the User's Master File and need be reset only when a new session is begun at a terminal with different communication facilities.

<<<:>>>

Defining Model of Tektronix Terminal--EQUIPMENT

Function: To set graphic parameters to be used to scale the display for specific Tektronix storage-tube terminal models.

Format: EQUIPMENT, k1

Notes: (a) The integer paramater k1 defines the Tektronix model for the terminal being used. The system default assumes that a model "4014" terminal is being used. Other recognized values for k1 are "4010" and "4051".

(b) This command must be entered prior to the Tektronix mode-setting command "INTERACTIVE, STORAGE", if it is to be in effect during the current session.

<<<:>>>

Set Program for Passive-Mode Operation--BATCH

Function: To specify that this session is to be a batch computer run.

Format: BATCH

Note: The operation mode can be set only once during a session or run.

The only entries which may precede the mode setting commands are the commands INITIAL, RESTART, and parameter-setting commands.

<<<:>>>

Set Program for Interactive-Mode Operation--INTERACTIVE

Function: To specify that this session is to be conducted from an interactive terminal and to define the type of terminal being used.

Format: INTERACTIVE, a1

Notes: (a) The alphanumeric parameter a1 may be:

STORAGE--to set the program in Tektronix storage-tube graphics mode.

TIMESHARING--to set the program in alphanumeric terminal (non-graphic) mode.

(b) The operation mode can be set only once during a session or run. The only entries which may preceed the mode-setting commands are the commands INITIAL, RESTART, and parameter-setting commands.

(c) If all commands and definitions are being entered from the keyboard during an INTERACTIVE, STORAGE session (the user has not provided a passive input file), an END command must be entered before the program will begin to prompt the user for new entries.

TERMINATION

The END command is used to terminate various phases of GPRIME processing. If there is no subordinate phase in process, an END command will terminate the session. The normal termination of a GPRIME session will leave the User Master File (UMF) data base in the proper state to permit the next session to continue as if there had been no interruption in processing.

General End Delimiter--END

Function: To indicate the end of a GPRIME processing phase, including the end of the current session.

Format: END

Note: The end-of-file mark on passive input files will also be recognized as an end delimiter.

Time Limit Termination

Function: To take an orderly exit from a GPRIME session when the system interactive time limit is about to be exceeded (for installations like DTNSRDC which have such a limit).

Note: When fewer than 20 seconds of processing time remain for an interactive session, GPRIME will begin issuing time limit warnings. If the session is still active when fewer than 10 seconds are remaining, GPRIME will attempt a normal termination, as if the user had entered an END command. Time limit tests are made between the processing of succesive commands and definitions. A request which requires a lot of processing can result in a termination without enough time for a normal exit. Therefore, heed the warning messages, exit, and begin a new session, or you may lose your data base.

CONTINUATION

RESTART is a reinitialization command for continuing GPRIME sessions and runs. It is the first command given and is the only way that a continuation session must differ from an initial session.

Continue Processing After a Break in a GPRIME Project--RESTART

Format: RESTART [,a1]

Note: The optional alphanumeric parameter a1 is used to select installation-dependent options, where necessary.

INTERACTIVE GRAPHICS EXAMPLES

The following sequence of commands will initiate a typical interactive graphics run on a CDC computer from a Tektronix storage-tube terminal:

```
GPRIME,MYDATABASE,MYID,NEW  -- CDC system directive
INITIAL
PRINT, ON                   -- Set printed-output flag
RATE, 30                    -- Graphic rate is not default
INTERACTIVE, STORAGE
END
```

After these commands have been processed, GPRIME will erase the graphics screen and prompt the user for further commands and definitions. GPRIME requires the following sequence of commands to begin a subsequent session from the same terminal:

```
GPRIME,MYDATABASE,MYID
RESTART
INTERACTIVE, STORAGE
END
```

USER HELP

GPRIME is not a self-teaching program. It does have an extensive user help feature which gives brief summaries of each of the commands and definitions. These summaries list all possible parameters associated with the commands and definitions and are available to assist interactive users. An alphabetical listing of all HELP messages can also be printed to obtain a concise, up-to-date reference manual.

Note that the HELP summaries are not contained in the GPRIME program, but reside in a separate data base called the System Master File (SMF). An updated SMF must be installed with each new version of the program, in order for these summaries to accurately reflect the current version of the program.

Interactive Command and Definition Summaries--HELP

Function: To display summaries documenting a particular GPRIME function.

Format: HELP [,a1] [,a2] [,a3]

Notes: (a) If no parameters are specified, GPRIME will list the broad categories under which the summaries are grouped. The four current categories are:

COMMANDS
DEFINITIONS
EXECUTIVE CONTROL
SYNTAX

These category names may be entered as the a1 parameter to obtain general summaries.

(b) If the user specifies either COMMANDS or DEFINITIONS as the a1 parameter, GPRIME will display a list of all current command names or definition names. Any of these names may be used as the a2 parameter to obtain a detailed summary for a particular command or definition.

(c) The command:

HELP, COMMAND, GGEN

will display a list of all possible commands for the GPRIME/GGEN interactive data generation program. The command names listed may be used as the a3 parameter to obtain a detailed summary for a particular GGEN command. (This summary information is also available from within the GGEN program, where the format of the HELP command is shortened to "HELP [,a3]".)

Print GPRIME Summary Information--DOCUMENT

Function: To Print All Current GPRIME Summary Information.

Format: DOCUMENT

Note: This summary information will be copied to GPRIME's normal printer file (the OUTPUT file on CDC computers). The user must provide for the proper disposition of that file at the end of the GPRIME session (e.g., printed at his terminal or routed to a high-speed printer).

DELETING AND CHANGING DEFINITIONS

GPRIME variables can be removed from the current problem by invoking the DELETE command. Similarly, GPRIME variables can be redefined by using the CHANGE command. Note that there is an important difference between a CHANGE redefinition and the deletion of a variable followed by a new definition having the same variable name. A CHANGED definition will cause all other variables dependent on the variable being changed to also be redefined appropriately. Deletion, followed by redefinition, causes only that one variable to be changed. If the user then wants the changes in the variable to be propagated to dependent variables, the CHANGE command can be applied to some or all of the variables dependent on the variable that was redefined.

Deleting Variables--DELETE

Function: To remove specified variables from the current data base User Master File (UMF).

Format: DELETE, a1, a2, <a3><k1>, THRU, <a3><k2>, a4, ...

Notes: (a) The alphanumeric parameters a1, a2, a4, etc. are the names of specific GPRIME variables that are to be deleted.

(b) When the "THRU" construction is used, all variables of the type indicated by the alphabetic string a3, having a numeric name between and including k1 and k2, will be deleted.

(c) The GPRIME processor ignores requests to delete undefined variables.

(d) The regions of the User Master File (UMF) occupied by deleted variables will be reused for subsequent definitions. However, deletions will not reduce the size of the current UMF.

<<<:>>>

Redefining Variables--CHANGE

Function: To change the definition of a GPRIME variable and all variables dependent on that variable.

Format: CHANGE, a1 [,<new variable definition>]

Notes: (a) The alphanumeric parameter a1 is the name of the variable to be redefined.

(b) The parameter a1 and <new variable definition> must form a valid definition statement for the specified variable. If <new variable definition> is omitted, the current definition will be updated to reflect any changes that have been made to the User Master File.

DISPLAYING DEFINITIONS

The user may request to see the definition statement associated with any GPRIME variable by simply entering the name of that variable. Note that the definition will be displayed in the Variable Definition Table (VDT); thus, for the request to be honored, the VDT feature must be turned "ON". A complete list of all active definitions can be obtained via the BACKUP command (this listing is generated on the FEDATA file).

REPEATED DEFINITIONS

Several GPRIME commands are available to generate sequences of points. These commands cause point names to be generated and explicit point definitions to be entered in the User Master File.

Points That are Equally Spaced Along a Curve--GENERATE, ARCLENGTH

Function: To generate a sequence of points along a curve, where the points are equally spaced with respect to arc length.

Format: GENERATE, ARCLENGTH, Ci, k1

Notes: (a) A sequence of k1 points will be generated along the curve Ci. These points will divide the curve into segments of approximately equal arc length. The number of points requested, k1, must not be less than two.

(b) The current implementation of this command generates points from the end of the curve first and the proceeds toward the beginning (ordered by decreasing parameter value).

(c) For future reference the definition of each point generated is entered into the UMF in the form "Pk2, POINT, Ci, BPJ" (Parametric form). The

parameter k2 is selected by the processor to define a unique point name and the parameter variable BPj is defined to specify the parametric location of the point on the curve Ci.

<<<:>>>

Interactively Defined Points on a Curve or Surface--POINTS

Function: To interactively define a sequence of points along a curve or on a surface.

Format: POINTS [,a1], TC

Notes: (a) This command is the repetitive form of the point definition by graphic input (Pi, POINT [,a1], TC). Any error condition or the user's end-of-graphic-input signal (often the "E" key) will end the definition sequence.

(b) The optional parameter a1 is used to specify the name of a reference curve or surface.

(c) For future reference the definition of each point generated is entered into the UMF in the form "Pki, POINT, a1, BPj" (parametric form). The parameter k1 is selected by the processor to define a unique point name and the parameter variable BPj is defined to specify the parametric location of the point on the curve or surface.

COMMENTS

The GPRIME user can include comments at any point within a GPRIME session. Comments are helpful for documenting the function of macros and the function of long definition sequences which have been prepared as passive input. Comments are also useful for annotating the printed output file during an interactive session.

Commentary and Annotation--COMMENT

Function: To permit the user to include documentary information within a GPRIME command and definition sequence.

Format: COMMENT, <any text>

Note: Parentheses must be excluded from commentary text to avoid a conflict with in-line definition processing.

CONTROL PARAMETERS

The GPRIME user can override any of the default parameters which affect his project environment. Many of the parameters are related to the graphic display and are described later. The remaining control parameters influence the B-spline fitting procedures, govern the quantity of printed and diagnostic information, and specify the procedures for handling abnormal conditions.

Many of the parameters control binary options (e.g., yes/no, on/off). Each of these parameters may be set to either the value "ON" or the value "OFF". All binary options are defaulted to "OFF".

Mathematical Precision Parameters

Default B-Spline Fitting Parameter--FIT

Function: To establish the number of points and B-spline functions to be used to approximate classical curves and surfaces.

Format: FIT, k1 [,k2]

Notes: (a) The integer parameter k1 is used to specify the number of points to be used in approximating classical curves and surfaces (default value is 24). For classical curves (e.g., a circle), k1 points will be evaluated at uniform intervals along the curve. For classical surfaces (e.g., a sphere), a k1 by k1 mesh of uniformly spaced points will be evaluated over each surface. Forms of curves and surfaces which are known to be linear, will be evaluated only at the end points or edges, rather than at k1 points.

(b) The optional integer parameter k2 is used to specify the number of B-spline functions that will be used in approximating classical curves or sur- faces (default value is k1/2, truncated). Forms of curves and surfaces which are known to be linear will be approximated by two B-spline functions rather than by k2 functions.

(c) Acceptable values for k1 and k2 must be at least 4 and no greater than 32. k2 must not exceed k1.

(d) These parameters will also be used as default values for definitions having the number of points or the number of functions as optional parameters.

<<<:>>>

Acceptable Numerical Equality Parameter--TOLERANCE

Function: To establish a closeness parameter which will be used to define numerical equality of computed values.

Format: TOLERANCE, k1

Notes: (a) The integer parameter k1 is used to define numerical equality for use in GPRIME's intersection solvers and similar modules (default value is 4). This parameter will be used in tests involving the number of significant digits of agreement between two quantities. This parameter is also used in setting an iteration limit in the intersection solvers.

(b) Changing the value of this parameter can improve the success of intersection procedures which have not converged. However, such intersection failures are usually indicative of other problems and thus will not be affected by changes in this parameter.

FINITE ELEMENT DATA GENERATION

The primary purpose of developing a geometric language was to help automate the finite element generation process. Currently there are three different procedures for generating finite element data using GPRIME geometry. Shell- and plate-type data may be generated interactively using the GGEN program; three-dimensional, solid-type data can be generated using the SOLIDGEN program;[3] and user-written data generators can be constructed with the subroutines on the GPRIME B-spline library.

GGEN Interactive Data Generator

The GGEN generator is invoked from GPRIME by entering the command

GGEN [,WSi] [,<point, curve, surface, and group list>]

where the optional parameter WSi is the name of a GGEN "workspace" to be associated with the data to be generated, and the optional <point, curve, surface, and group list> specifies various GPRIME geometric elements to be used in data generation.

Documentation for the GGEN commands can be obtained using GPRIME's HELP feature. Usual GGEN modeling procedures include the automatic generation of triangular and quadrilateral element meshes over specified surfaces, the direct inclusion of GPRIME points, and the interactive modification of the automatically generated meshes. The generated finite element model can be copied to the file FEDATA by invoking GGEN's PUNCH command.

SOLIDGEN 3-D Solid Generator

The SOLIDGEN generator is run as a separate batch-mode job after the surfaces bounding the model have been defined and stored in a GPRIME data base (User Master File). Along with the geometric data, modeling control information will be required for the SOLIDGEN run, including specification of the type and number of elements to be generated.

User-Written Data Generators

User-written data generators may be interactive- or batch-mode programs which reference previously defined GPRIME geometric information (as described in the GPRIME DATA BASE section of this manual). Subroutines from the GPRIME B-Spline Library can be used to evaluate points on curves and surfaces, to fit curves and surfaces to collections of points, and to find intersections between curves and surfaces. Library subroutines can also be used to evaluate sets of points according to a specified criterion (e.g., uniform arc-length spacing along a curve).

MACRO FACILITY

The GPRIME macro capability permits the user to define a sequence of commands and definitions which can later be invoked as one command. Macros may be defined to have substitutable parameters as arguments, permitting the same sequence of commands and definitions to be executed with different sets of parameters, each producing different constructions. These arguments may be used in any command or definition within the macro.

Commands are available to create, execute, list, change, and remove macros. A PAUSE command, which temporarily halts processing, is available for use within a macro sequence. COMMENT commands should be liberally included within macros to document particular functions and restrictions. A listing of all current macro definitions can be obtained with the BACKUP command.

The GPRIME macro facility is quite primitive when compared with the features that are currently available in macro compilers for computer languages. This is an area for further GPRIME development, but only the features described in this section are currently available.

Create a Macro Sequence--BUILD

Function: To create a sequence of commands and definitions which can be executed as a single command.

Format: BUILD, a1 [,k2]

Notes: (a) The alphanumeric string a1 is the name of the macro to be created. Future references to this macro must include the complete name as defined (no three-character abbreviations).

(b) The optional integer parameter k2 specifies the number of substitutable parameters which will be used in this macro definition. The default is no substitutable parameters. The maximum number of parameters is 23. Within the text of the macro entry, parameters are referenced by the symbols: +1, +2, ..., +23. The number which follows the plus sign indicates the number of the referenced parameter in the list supplied when the macro is executed.

(c) This command establishes the macro creation mode and allows the user to enter lines of text to be stored in the macro. During macro creation, a macro entry can be deleted by entering the line number of the line to be deleted. Macro entries can be replaced by beginning the replacement line with the number of the line to be replaced. To exit the macro creation mode, type END.

(d) Individual macro definitions may be no longer than 25 lines; however, multiple statements can be included on a single line by using a semicolon as a delimiter. The last statement of a macro sequence may be an EXECUTE command invoking a follow-on macro. This technique can be used to create arbitrarily long macros and also to define recursive procedures (macros can reinvoke themselves).

(e) In-line definitions are not permitted within a macro.

<<<:>>>

Execute a Macro Sequence of Commands and Definitions--EXECUTE

Function: To execute a macro sequence of commands and definitions, with optional substitutable parameters.

Format: EXECUTE, a1 [, c2, c3, .. , ci]

Notes: (a) The alphanumeric string a1 is the name of the macro to be executed. The name must be entered exactly as defined in the BUILD command used to create the macro.

(b) The character parameters, c2, c3, ..., ci, are character strings which will be substituted throughout the macro sequence as indicated in the macro definition. In general, substitutable parameters are optional, but each macro execution must have the same number of parameters as specified in the definition.

<<<:>>>

Display a Macro Definition--LIST

Function: To list a copy of a macro definition at the terminal and the standard output file.

Format: LIST, a1

Note: The alphanumeric string a1 is the name of the macro to be listed. The name must be entered exactly as defined in the BUILD command used to create the macro definition.

<<<:>>>

Change an Existing Macro Definition--EDIT

Function: To add, replace, or delete lines within a macro definition.

Format: EDIT, a1

Notes: (a) The alphanumeric string a1 is the name of the macro to be changed. The name must be entered exactly as defined in the BUILD command used to create the macro definition.

(b) New lines may be appended to a macro by entering those lines in sequence without line numbers.

(c) Lines may be replaced by beginning the replacement line with the line number of the line to be replaced.

(d) Lines may be deleted by entering a line which contains only the line number of the line to be deleted. Note that deletion of lines causes any following lines to be renumbered.

(e) To exit the macro editing mode, type END.

<<<:>>>

Remove a Macro Definition From a User's Master File--PURGE

Function: To remove a macro definition from the User Master File associated with the current project.

Format: PURGE, a1

Note: The alphanumeric parameter a1 is the name of the macro to be removed. The name must be entered exactly as defined in the BUILD command used to create this macro.

<<<:>>>

Halt Execution Within a Macro Sequence--PAUSE

Function: To temporarily halt the execution of a macro sequence.

Format: PAUSE

Notes: (a) This command permits the user to decide whether macro execution should continue or be halted. Macros can be invoked recursively (the macro ends with an EXECUTE command with its own name) and the PAUSE command gives the user the capability to intervene in this process. Suspended macro execution is permanently halted by typing END. Any other entry will cause the execution to be resumed.

(b) The PAUSE command is valid only within a macro definition.

<<<:>>>

Macro Definition and Execution Example

Macros are most helpful in performing tasks that are cumbersome to type in at the terminal. The need to display only a few geometric elements occurs frequently. The following macro will display three or fewer elements:

```
BUILD, SHOW3, 3
HIDE, ALL
PLOT, +1, +2, +3
DISPLAY
PLOT, ALL
END
```

To display surfaces S1, S2, and curve C3 the user would enter:

```
EXECUTE, SHOW3, S1, S2, C1
```

Note that the next DISPLAY command entered will display all geometric elements. Because the number of substitutable parameters at execution must agree with the number of parameters specified in the definition, the following entry would be required to display only surface S1:

```
EXECUTE, SHOW3, S1, S1, S1
```

CALCULATOR MODE

A rudimentary interactive calculator is available to GPRIME users. Sequences of simple arithmetic operations can be performed to assist in geometric definition.

Enter Calculator Mode--MATH

Format: MATH

Note: Current result initialized to zero.

Add a Real Constant to the Current Result--"+"

Format: +r

Note: r is any real constant.

Subtract a number from the Current Result--"-"

Format: -r

Multiply Current Result by a Real Constant--"*"

Format: *r

Divide Current Result by a Real Constant--"/"

Format: /r

Raise the Current Result to a Real Constant Power--"'"

Format: 'r

Save Current Result--"S[AVE]"

Format: SAVE

Use Previously Saved Result in a Calculation--"R[ESTORE]"

Format: a1 RESTORE

Note: The character constant "a1" must be one of the five arithmetic operators; +,-,*,/,or'.

Use the Current Result Twice in Calculation--"L[AST]"

Format: a1 LAST

Terminate Calculator Mode--"E[ND]"

Calculator Mode Example

The following sequence calculates the length of a hypotenuse of a right triangle with sides of length 5.5 and 21.5:

```
MATH
+5.5
 *L
 S
```

-L

+21.5

*L

+R

'.5

E

GRAPHIC DISPLAY

GPRIME's graphic display capability includes item selection, control of viewpoint, vantage point, scaling, annotation, rotation about axes, hidden line removal, reference tables, and program-to-user messages. Each of these features may be selected or disabled by the user, as required.

Screen Format

Figure 5 (p.12) illustrates the general layout of the GPRIME display screen. This display was made with the screen-size option set to "LARGE". In this mode geometric elements and reference information both appear within the large square box, which nearly fills the screen. This mode provides the maximum display area, but interference between text and geometry plots can occur. The screen-size option "SMALL" specifies a smaller geometric plotting square with all reference information displayed outside that square.

GPRIME always displays a "TIME LEFT" message at the top of the screen on those systems which have a processing time limit. Plots can be marked with a time and date display. Item A in Figure 5 (IDENTIFY). A chronological listing of the names of all variables defined in the current session can be shown, Item B. (This table is known as the Variable Name Table--VNT.) Definition statements can be displayed as they are processed, Item C. (This table is known as the Variable Definition Table--VDT.) The current status of all plotting options and display parameters can be listed in the STATUS table, Item D. A set of AXES can be displayed at the right of the screen to help orient the viewer.

Prompting lines will always be displayed when GPRIME expects data to be entered during processing in an interactive mode, Item E. These prompting lines are 80 characters long, the maximum length .epted by GPRIME. Variable

name annotation can be added to all viewable geometric elements (LABEL). Messages generated by the GPRIME processor will be displayed in the lower right corner of the screen.

<<<:>>>

Reference Axes Option--AXES

Function: To display base coordinate system axes for reference.

Format: AXES, a1

Note: If the alphanumeric parameter a1 is specified as "ON", the reference axes will be displayed immediately and redisplayed whenever the screen is replotted. Invoking AXES with a1 set to "OFF", prevents further redisplay of the reference axes.

<<<:>>>

Time and Date Option--IDENTIFY

Function: To set a plot parameter which controls the display of the current time and date at the top of the screen.

Format: IDENTIFY, a1

Note: If the alphanumeric parameter a1 is set to "ON", time and date information will be included when the display is replotted. If a1 is set to "OFF", the time and date will not be displayed.

<<<:>>>

Variable-Name Annotation-- LABEL

Function: To set a plot parameter which controls the labeling of geometric elements.

Format: LABEL, a1

Note: If the alphanumeric parameter a1 is set to "ON", the display of each geometric element will be accompanied by a variable-name label. When a1 is set to "OFF", no name labels will be displayed.

<<<:>>>

Display Size Option--SCREEN

Function: To select the size of the geometric plotting area on the screen.

Format: SCREEN, a1

Note: The alphanumeric parameter a1 must be either LARGE (default) for

full screen geometric plotting or SMALL for reduced-size, no-interference plotting.

<<<:>>>

Status of Current Plotting Parameters Display--STATUS

Function: To display the current status of various GPRIME plot and control parameters.

Format: STATUS [,al]

Notes: (a) If the alphanumeric parameter al is omitted or specified as "ALL", all plotting control parameters will be displayed.

(b) If al is set to "BSPLINE", the GPRIME TOLERANCE and default mesh size parameters will be displayed.

(c) If al is set to "ENVIRONMENT", the currently defined plot subset will be displayed (this is the set defined by the HIDE, PLOT, POINTS, CURVES, and SURFACES commands).

(d) If al is specified as "FLAGS", all GPRIME control parameters will be displayed (e.g., AXES, LABELS).

(e) If al is specified as "GEOMETRY", current VIEW command parameters will be displayed.

<<<:>>>

Variable Definition Table Display Option--VDT

Function: To set a parameter which controls the display of the Variable Definition Table.

Format: VDT, al

Note: If the alphanumeric parameter al is set to "ON", the Variable Definition Table will be displayed when the screen is redrawn. If set to "OFF", the table will not be displayed.

<<<:>>>

Variable Name Table Display Option--VNT

Function: To set a parameter which controls the display of the Variable Name Table.

Format: VNT, al

Note: If the alphanumeric parameter al is set to "ON", the Variable Name Table will be displayed when the screen is redrawn. If set to "OFF", the table will not be displayed.

Redisplay of Program Message--REPEAT

Function: To redisplay active screen contents and the last message issued by the program.

Format: REPEAT

Note: If the storage-tube screen is filled with information, the message area of the screen may be cluttered and additional program messages may be unreadable. The "REPEAT" command is used to redisplay currently active screen contents and the last program message.

Viewing Options

The GIRIME graphic display is a three-dimensional, perspective plot of a region of geometric space which has been selected by the user. The region is defined by specifying a vantage point (location of the field of view), and an eye-to-screen distance. The relationship between these parameters is shown in Figure 11. The viewing parameters may be changed at any time and they may be modified by scale factor and rotations, as well. Plots of any view can be made with hidden lines removed.

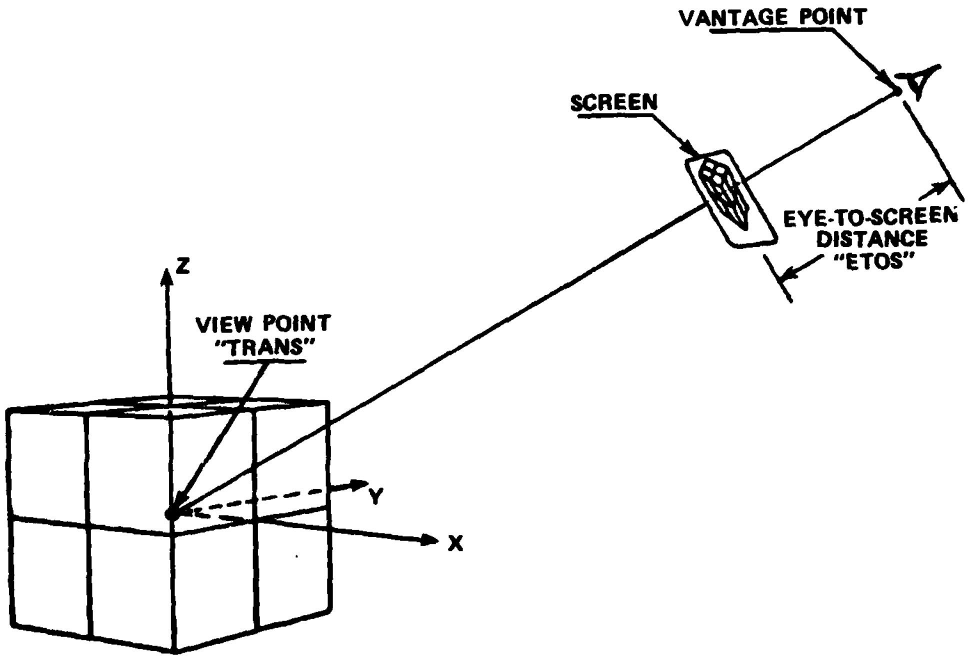


Figure 11 - View Parameters

GPRIME's default view has the view point located at the origin of the base coordinate system, the vantage point located at (20,20,20) in that system, and an eye-to-screen distance of 8. In this view, a 20 unit cube centered at the origin nearly fills the screen.

Plot Display with New Viewing Parameters--VIEW

Function: To remove the previous display from screen and replot it with new viewing parameters.

Format: View [,EYE,r1,r2,r3] [,TRANSLATE,r4,r5,r6] [,ETOS,r7]
[,MODIFY,r8] [,CLIP,k1]

Notes: (a) The VIEW command, with no optional parameters, plots the current display subset using unmodified display parameters. Its effect is identical to the DISPLAY command.

(b) When included with the EYE keyword, the real parameters r1, r2, and r3 define the coordinates of the vantage point (location of viewer's eye).

(c) When included with the TRANSLATE keyword, the real parameters r4, r5, and r6 define the coordinates of the view point (the center of the field of view).

(d) When included with the keyword ETOS, the real parameter r7 defines the eye-to-screen distance (see Figure 9). This parameter seems awkward and "unnatural" to most users. Its effect is a scaling of the entire plot, which can be obtained more "naturally" with the MODIFY parameter.

(e) When included with the keyword MODIFY, the real parameter r8 defines a scale factor to be applied to the plot defined by the currently defined vantage point, view point and eye-to-screen distance. Note that the effects of this scaling parameter are not cumulative. Thus, a plot defined by a particular view command will be exactly reproduced by subsequent invocations of that view command.

(f) When the keyword CLIP is included with the VIEW command, the plot will be made with hidden lines removed. The optional integer parameter k1 defines a level resolution for the hidden lines removal processing (see the CLIP command). The default value for k1 is 32.

(g) Currently, GPRIME does not give the user control over the orientation of the plot. The basic z-axis will always be oriented vertically,

positive upward, on the screen. When looking along the z-axis, the y-axis will be vertical on the screen.

<<<:>>>

Plot Current Display--DISPLAY

Function: To remove the previous display from the screen and replot it using currently defined viewing parameters.

Format: DISPLAY

Note: This command plots the current display subset using unmodified viewing parameters. Its effect is identical to a VIEW command with no parameters specified.

<<<:>>>

Erase the Terminal Screen--WIPE

Function: To remove the previous display from the screen.

Format: WIPE

<<<:>>>

Rotate the Observation Point--ROTATE

Function: To incrementally change the viewing parameters by rotating the observation point about the view point.

Format: ROTATE, r1, a2

Notes: (a) The real parameter r1 defines the incremental rotation of the vantage point about a line through the view point which is parallel to the coordinate axis specified by the a2 parameter (rotations follow the "right hand rule"). The display will be replotted with the new viewing parameters. This procedure will be repeated with additional rotational increments made to the viewing parameters. The program will pause between plots. At that time the user may enter "END" to terminate the sequence. Any other entry will cause the next increment to be processed.

(b) The alphanumeric parameter a2, which defines the axis of rotation, must be "X","Y" or "Z"

Display of Selected Items

It is frequently necessary to limit the GPRIME display to a few of the defined items. Several mechanisms are available for partitioning the geometric elements into plottable subsets. These include direct specification of

items which are to be plotted and items which are not to be plotted. Classes of items, like CURVES, can be declared plottable or not plottable. The following commands set parameters and display nothing directly, but they influence subsequent VIEW, DISPLAY, CLIP, ROTATE commands.

Declare a Geometric Element Plottable--PLOT

Format: PLOT [,ALL] [,<list of variables name>]

Notes: (a) The PLOT,ALL command resets the display selection parameters so that all items will be selected for plotting.

(b) Several PLOT commands with lists of item, can be used to include large subsets.

<<<:>>>

Declare a Geometric Element Not Plottable--HIDE

Format: HIDE [,ALL] [,<list of variable names>]

Notes: (a) The HIDE, ALL command resets the display selection parameter so no items will be selected for plotting.

(b) Several HIDE commands, with lists of items, can be used to exclude large subsets.

<<<:>>>

Include or Exclude All Points from Plottable subset--POINTS

Format: POINTS, a1

Notes: (a) The alphanumeric parameter a1 may be set to ON to include all points in the plottable subset or set to OFF to exclude all points.

(b) Points explicitly selected via HIDE or PLOT commands will not be affected.

<<<:>>>

Include or Exclude All Curves from Plottable Subset--CURVES

Format: CURVES, a1

Notes: (a) The alphanumeric parameter a1 may be set to ON to include all curves in the plottable subset or set to OFF to exclude all curves.

(b) Curves explicitly selected via HIDE or PLOT commands will not be affected.

<<<:>>>

Include or Exclude All Surfaces from Plottable Subset--SURFACES

Format: SURFACES, a1

Notes: (a) The alphanumeric parameter a1 may be set to ON to include all surfaces in the plottable subset or set to OFF to exclude all surfaces.

(b) Surfaces explicitly selected via HIDE or PLOT commands will not be affected.

<<<:>>>

The display subset in effect at any given time can be viewed as the cumulative result of all the display selection commands encountered since the last HIDE, ALL or PLOT, ALL command (parameters are initially defaulted to have all items declared plottable). Typical selection sequences might include the following commands:

HIDE,ALL; PLOT,S1,S2; CURVES,ON; POINTS,ON

PLOT,ALL; CURVES,OFF; HIDE,S1,P17,P18

Frequently required subsets may be selected via a macro sequence made up of HIDE and PLOT commands.

Plots with Hidden Lines Removed

GPRIME geometric displays are usually drawn as if all surfaces were transparent; that is, all lines are drawn. Any display can be drawn showing only those features that would be visible if all surfaces were considered opaque. For these plots GPRIME uses an approximate hidden line removal technique which sub-divides the screen into a gridwork of small picture elements, or pixels. The visibility of line segments can be determined only to within the width of one pixel. Thus the more pixels, the better the quality of the hidden line removal. However, more pixels require more computer memory and more processing time. As the size of the pixels increases, plot quality is degraded because surfaces become partially transparent to nearly obscured features. The CDC version of GPRIME has enough memory to process hidden line removed displays with a 150 x 150 pixel gridwork. When this maximum is selected, display processing requires about twice the time necessary for a plot without hidden line removal. All hidden line removed plots in this manual have been made with this maximum gridwork.

<<<:>>>

Plot Current Display With Hidden Lines Removed--CLIP

Function:

Format: CLIP [,k1]

Notes: (a) The optional integer parameter k1 specifies the number of pixels to be used along the longest screen dimension. Default value is 32.

(b) Hidden line removal can be requested only for the current display. It cannot be specified for future plotting.

(c) Hidden line removal can be specified along with changes to plotting parameters by including CLIP at the end of a VIEW command.

PRINTING AND DIAGNOSTIC PARAMETERS

When requested, GPRIME will transmit a copy of all the commands and definitions processed to the standard printer file (the file OUTPUT on CDC computers). Copies of all error messages are automatically routed to this file, and users can request that various types of diagnostic information also be sent to the printer file. The interactive user must make provisions for the disposition of this file at the end of his GPRIME session.

<<<:>>>

Diagnostic and Immediate Error Termination Flag--DEBUG

Function: To set a parameter which causes many GPRIME modules to print diagnostic information for tracking down system errors and causes the program to stop immediately if such a system error occurs (skipping the normal closing procedures for the User Master File).

Format: DEBUG, a1

Notes: (a) The alphanumeric parameter a1 may be set to "ON" or "OFF".

(b) The type of diagnostic message varies from module to module. Such messages are designed for solving system-type problems and will not usually be much help in tracking down user's errors.

(c) The immediate termination feature is necessary to obtain a valid system dump after a system-type error.

<<<:>>>

Printing of User Master File Data Base--DUMP

Function: To copy the contents of the User Master File (UMF) base to the standard printer file.

Format: DUMP [,a1]

Notes: (a) If the optional parameter a1 is omitted, an immediate print-file dump of the UMF will be generated when this command is executed. If the

AD-A119 923 DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC F/G 9/2
GPRIME: A GEOMETRIC LANGUAGE FOR FINITE ELEMENT MODELING PROGRA--ETC(U)
SEP 82 J M MCKEE, M E GOLDEN, D R WALLACE
UNCLASSIFIED DTNSRDC-82/062 NL

2 of 2

END
DATE
FILMED
11 82
DTIC

al parameter is specified as "ON", an internal parameter will be set causing dumps of the UMF to be generated whenever the UMF is opened or closed.

(b) Execution of this command, with the al parameter specified as "OFF", will set the internal parameter to prevent further open/close dumps of the UMF.

(c) UMF dumps are system debugging aids. They usually require several hundred pages of paper for printing and are not useful for identifying user's errors.

<<<:>>>

Print a History of All User's Transactions--PRINT

Function: To set a parameter that will invoke the printing of a record of all explicit and implicit user transactions during a run or session.

Format: PRINT, al

Notes: (a) The alphanumeric parameter al may be set to "ON" or "OFF".

(b) The transaction history controlled by this command is very useful in tracking down user's errors and identifying the causes of unexpected events.

<<<:>>>

Print Timing Information--TIME

Function: To set a parameter that will invoke the printing of starting and completion times to be printed for the execution of each GPRIME command and definition.

Format: TIME, al

Notes: (a) The alphanumeric parameter al may be set to "ON" or "OFF".

(b) Timing information is most useful for system development and is seldom directly helpful to the user.

<<<:>>>

Print a History of User Master File--TRACE

Function: To set a parameter for printing diagnostic information for each UMF transaction.

Format: TRACE, al

Notes: (a) The alphanumeric parameter al may be set to "ON" or "OFF".

(b) Printouts of UMF transaction data are a system debugging aid. They are not useful for identifying user's errors.

SYMBOLIC DATA BASE ARCHIVAL

Information is stored in a GPRIME data base in binary form. This form was chosen for computational efficiency; however, it is only program readable (it can't be edited by a text editor, for example). Two types of character-format data can be obtained from GPRIME for archival purposes. The user-definition type of backup includes the user's typed-in line for all current geometric definitions (or equivalent for graphic input). The B-spline type of backup includes the internal B-spline representation for all current geometric definitions. Both types include symbolic listings of current macro definitions. These back-up definitions can be used directly to create a new equivalent data base.

The backups do not include parameter information, so appropriate fitting, plotting, and viewing specifications must be added to produce a data base that is identical to the original.

To make the user's symbolic data useful, every referenced item must be available at the time of the backup; that is, these items can not have been deleted. Otherwise, some variables will not be properly defined. A B-spline-format definition does not reference any other variable, so such backups will always yield properly defined variables (at the cost of eliminated cross referencing). A combined format (user's definitions followed by the B-spline definitions) can be used to recreate a data base with as much cross-referencing as the original (at the cost of an error message for each doubly defined item).

<<<:>>>

Archive Data Base Information--BACKUP

Function: To produce a symbolic listing of the current data base file (UMF).

Format: BACKUP [,a1]

Notes: (a) If the alphanumeric parameter a1 is omitted or specified as "USER", the archive listing will be in the user type-in format. If a1 is specified as "BSPLINE", the listing will be in the internal B-spline format. If a1 is specified as "ALL", the combined format will be produced.

(b) Archival listings are written to GPRIME's punched output file, FEDATA.

COMMAND INDEX

In this CONTROL OF PROCESSING section, the GPRIME command descriptions have been grouped into functional categories. This index references commands by their basic category. Page numbers for the basic categories are listed in the TABLE OF CONTENTS.

THE GPRIME DATA BASE

Ordinarily the GPRIME user is not concerned with his GPRIME data base. The data base resides on the User Master File (a file named UMF) and is available for restarting GPRIME after one or more sessions or runs. If a user has a very large data base or if he needs to access his GPRIME geometric definitions from other programs, a knowledge of the data base structure becomes necessary.

DATA BASE FORMAT

All of the user's environmental parameters, definitions, macro definitions, and GGEN-generated finite element data are stored in the UMF established for that problem. Geometric definitions make up the bulk of a user's data base; because users may wish to access this data directly in their own programs (to create special purpose data generators, for example), the following detailed discussion of the structure and format of definition data has been included.

Each entry in the data base is identified by a key word which is included as part of that entry. For geometric definitions, the variable name associated with the stored definition is used as a component of the key word for that data. Different types of data associated with a particular variable are stored as subgroups under that variable name and the subgroup name also becomes a component of the key word associated with a data entry (the key word identifying the user's typed-in definition of surface S1 would be S1S2D and the key word identifying the internal B-spline definition of that surface would be S1S4D). If there are several entries for a particular type of data (for example, the curves drawn for visual presentation of a surface), an integer sequence number is appended to the associated key word. For each of these data entries (the "D" in the key word) in the data base, there is an associated length entry (two words long, key word and integer length word) which has a key word which is the same as the data entry, except that the "D" is replaced by an "L" (e.g., S1S4L). Definitions of GPRIME's 13 subgroup types are contained in Table 2.

TABLE 2 - GPRIME SUBGROUP DEFINITIONS

SUBGROUP KEY	CONTENTS OF SUBGROUP DATA
S1D	Variable type
S2D	User definition
S3D	Object coordinate space contours for plotting
S4D	Parametric form of definition
S5D	Screen coordinate space contours for plotting
S6D	Cross-reference table of dependent variable names
S7D	Interactive display item pointers for refresh graphics
S8D	Error conditions and messages
S9D*	Properties loads and constraint data
S10D*	Analysis response request
S11D*	Topological data
S12D*	Analysis input data
S13D*	Analysis response requests

* These subgroups have been defined but are not currently used by GPRIME.

There are several ways to locate GPRIME data, but access to definition data is most efficient when the Variable Pointer Table is used. Each entry in this table, called a Variable Pointer Record (VPR), contains pointers to all the subgroups associated with one particular variable. Table 3 defines a typical VPR.

TABLE 3 - VARIABLE POINTER RECORD (VPR) STRUCTURE

WORD INDEX	CONTENTS
1	Pointer to previous VPR (zero if first VPR).
2	Variable Name (left justified, zero filled).
3	Disk address of Subgroup 1 data (left half of word). Length of Subgroup 1 data (right half of word).
i	Address and length data for Subgroup i-2.
15	Disk address of Subgroup 13 data (left half of word). Length of Subgroup 13 data (right half of word).
16	Pointer to next VPR (zero if last VPR).

ACCESS FROM OTHER PROGRAMS

A set of user callable subroutines is available for user access to the internal B-spline definitions stored on a UMF data base. These subroutines are referred to as the GPRIME Data Base Access Capability. Their functions are to: (1) open the UMF for reading, (2) locate definitions by searching the Variable Pointer Table, (3) copy the definitions from disk storage to the user's working storage, and (4) close the UMF file. (GPRIME's basic B-Spline subroutine library[1] is available for operations on this data from within a user's program.) If additional types of data are required by the user, for example, the user's typed-in definition, similar subroutines could easily be constructed simply by substituting appropriate subgroup references in the existing routines.

In order to retrieve basic geometric information from the UMF, the applications programmer must provide working storage in the form of two COMMON blocks. The first COMMON block, /GPRSRF/, is needed to hold the definitions in memory when they have been fetched from disk. The second, /GPRIDX/, is an index area which is used to permit access to several GPRIME entities at one time. The minimum length of /GPRSRF/ is determined by summing the length of the definitions that must be simultaneously available and adding one for each

such definition. When several groups of geometric definitions are alternately required, the performance of the access capability will be improved if /GPRSRF/ is large enough to contain all the definitions. The minimum length of /GPRIDX/ is nine times the number of definitions which must be available simultaneously. Disk searching time will be minimized if the length of /GPRIDX/ is nine times the total number of different definitions that will be accessed during the course of program execution.

To obtain definitions from a UMF the following sequence of subroutines must usually be called:

(1) GPRINT Open the data base file and initialize the capability

(2) GPRSET Specify the number of operands currently required

(3) GPRTST (optional) Test for the existence of a particular variable

(4) GPRGET Access data

(5) GPREND Close data base file

The appropriate UMF file must have been associated with the user's program (using the ATTACH,UMF command on CDC computers) before these subroutines can be executed.

Subroutine GPRINT

Function: To initialize the GPRIME geometry access capability.

Calling sequence: CALL GPRINT (LSURFS,LINDEX,IFAIL)

where

Argument (Attributes)	Contents
LSURFS (integer/input)	Length of /GPRSRF/
LINDEX (integer/input)	Length of /GPRIDX/
IFAIL (integer/output)	Flag word. A zero value of IFAIL indicates a successful completion. A value of 1 is returned if the date base does not exist or is not in the proper format.

<<<:>>>

Subroutine GPRSET

Function: To set the size of a new group of geometric definitions.

Calling sequence: CALL GPRSET (NGROUP,IFAIL)

where

Argument (Attributes)	Contents
NGROUP (integer/input)	Number of GPRIME definitions to be kept active in new group.
IFAIL (integer/output)	Flag word. A zero value of IFAIL indicates a successful completion. A value of 1 is returned if COMMON block /GPRIDX/ is too short to accommodate this group.

Notes: (a) Each time a new group of geometric definitions is needed, subroutine GPRSET must be called with the number of definitions to be encountered in the new group.

(b) If several groups consist of only one definition, all calls to GPRSET for those groups may be omitted except the first. A call to either subroutine GPRTST or subroutine GPRGET adds one member to the current group, even when one or more calls have been made previously with the same name. If the user attempts to have more than NGROUP definitions active, only those definitions set up by the last NGROUP calls are guaranteed to be valid (others may or may not be, depending on the available memory).

<<<:>>>

Subroutine GPRTST

Function: To determine whether a GPRIME definition exists in a data base and to determine the characteristics of that definition.

Calling sequence: CALL GPRTST (NAME,IFOUND,LEN,M1,N1,IGTYP,IBTYP,IFAIL)

where

Argument (Attributes)	Contents
NAME (alpha/input)	GPRIME name of definition to be accessed, in "H" or blank filled format.
IFOUND (integer/output)	Index address of specified definition. If zero, definition is not in data base.

LEN (integer/output)	Number of words required for this definition.
M1,N1 (integer/output)	Dimensions of B-spline definition if definition is for curve or surface. Set to zero if definition is not a curve or surface.
IGTYP (integer/output)	GPRIME type code. See Table 4 for definitions of GPRIME codes.
IBTYP (integer/output)	B-spline type indicator for curve and surface definitions. See Table 5 for definitions of B-spline codes. Set to zero if definition is not a curve or surface.
IFAIL (integer/output)	Flag word.

Value	Meaning
0	Successful completion.
1	Bad data base.
2	Not enough index space.

Note: Subroutine GPRTST is used only when GPRIME operands are to be checked early in a procedure, usually before any significant computations are performed.

TABLE 4 - GPRIME VARIABLE TYPE CODES

IGTYP	VARIABLE TYPE	IGTYP	VARIABLE TYPE
1	Arc	18	Scalar
2	Circle	19	Sphere
3	Cone	20	Surface
4	Coordinate System	21	Vector
5	Curve	22	Volume
6	Cylinder	23	Wedge
7	Ellipse	24	Orientation Transformation
8	Gconic	25	Fit
9	Group	26	Evaluate
10	Hemisphere	27	Opposite
11	Hyperbola	28	Offset
12	Intersection	29	Frustum
13	Line	30	X surface
14	Parallelepiped	31	Y surface
15	Plane	32	Z surface
16	Point	33	Math
17	Rename		

TABLE 5 - B-SPLINE CURVE AND SURFACE TYPE CODES

IBTYP	CURVE OR SURFACE TYPE
1	Open Curve
2	Closed Curve
3	Open Surface
4	s-Closed Surface
5	t-Closed Surface
6	s- and t- Closed Surface

<<<:>>>

Subroutine GPRGET

Function: To fetch a GPRIME geometric definition from the data base.

Calling sequence: CALL GPRGET (NAME, IS, LEN, M1,N1, IGTYP, IBTYP,IFAIL)

where

Argument (Attributes)	Contents
NAME (alpha/input)	GPRIME name of definition to be accessed, in "H" or blank filled format.
IS (integer/output)	Pointer to index /GPRIDX/ word which contains the pointer to the definition in /GPRSRF/.
LEN (integer/output)	Number of words required for this definition.
M1,N1 (integer/output)	Dimensions of B-spline definition if definition is for curve or surface. Set to zero if definition is not a curve or surface.
IGTYP (integer/output)	GPRIME type code. See Table 4 for definitions of GPRIME codes.
IBTYP (integer/output)	B-spline type indicator for curve and surface definitions. See Table 5 for definitions of B-spline codes. Set to zero if definition is not a curve or surface.
IFAIL (integer/output)	Flag word.

Value	Meaning
0	Successful completion.
1	Bad data base.
2	Not enough index space.
4	Not found.
5	Not enough space for definition.
6	Access program logic error.

Notes: (a) The ISth word of /GPRIDX/ always contains the subscript of the first word of the definition in /GPRSRF/. Calls to GPRGET may change the location of the definition (not IS). Hence, the programmer should update pointers to definitions when GPRGET has been called (for example, if ISUBSC is to be used as a subscript to the definition, referenced by IS in the SURFS array, ISUBSC = IDXGPR(SI).

(b) Subroutine GPRGET calls subroutine GPRTST to check the validity of the current request making independent calls to GPRTST unnecessary.

<<<:>>>

Subroutine GPREND

Function: To terminate all activity on the geometric portion of the data base.

Calling sequence: CALL GPREND

Subsequent access to GPRIME geometric information must begin with a call to subroutine GPRINT.

Implementation Dependent Subroutines

Subroutines GPRFIL, GPRHDR, GPREAD, RERROR, GPGRNM, and GPNTST are lower level subroutines which support the subroutines just described. Each of these subroutines has some features that must be tailored to the host computer and the data base access methods used. With the possible exception of GPNTST, these subroutines should not be of interest to the applications programmer.

<<<:>>>

Subroutine GPNTST

Function: To test the alphabetical characters that are used as the first part of GPRIME names.

Calling sequence: CALL GPNTST (NAME,NNAME,CHARS,NCHAR,IFAIL)

where

Arguments (Attributes)	Contents
NAME (alpha/input)	GPRIME name to be tested.
NNAME (alpha/output)	Left-justified GPRIME name.
CHARS (alpha/input)	Characters used to test name.
NCHAR (integer/input)	Number of characters in CHARS.
IFAIL (integer/output)	Flag word.

Value	Meaning
0	Successful match.
1	Failure.

Note: Subroutine GPNTST ignores any blank characters which precede the GPRIME name, but the word CHARS must not have any leading blank characters.

COMPUTER SYSTEM INTERFACE

The design philosophy used in developing GPRIME calls for computer-independent coding of the program. At some point, however, the program must interface with the host computer system. From the user's viewpoint, the current interface for CDC computers is restricted to a few control directives and one file access. The control directives are expanded into a complete set of directives for managing files and setting up the environment for interactive graphics processing. On other systems, similar interfaces will be required.

GPRIME FILES

All files used by GPRIME are listed in Table 6. Some are special random access files, some are required for the graphic interface, and others are standard FORTRAN sequentially accessed files. A GPRIME session will always require two random access disk files: (1) The User Master File (CDC local file UMF) and (2) the System Master File (SMF). The User Master File is the user's data base for the current problem. This file is created during an INITIAL session and that file, or a copy of it, must be available for each RESTARTED session. (We recommend using a copy of the previous UMF for subsequent sessions, since it provides a measure of security for previously stored data which could be accidentally changed during a subsequent session.) The System Master File contains user help information and GPRIME error messages. Both the UMF and SMF file may be copied to tape or disk files following a GPRIME session, but they must reside on a disk device to be accessed by GPRIME. For big problems the UMF file will grow quite large. Thus, copies of the data base should be discarded when they are no longer needed.

Five sequentially-accessed FORTRAN files may be used during a GPRIME session. These files are listed with a short description of usage in Table 6.

Two of the four Tektronix files are used with a given version of the program, one for graphic input and one for output. File TAPE 61 must be associated with the user's keyboard for "INTERACTIVE, STORAGE" mode runs.

TABLE 6 - GPRIME FILES

FILE NAME	FORTRAN FILE NUMBER	POSITIONING STATUS	USAGE
UMF	None	Always positioned by program	User's problem data base
SMF	None	Always positioned by program	System help and message file
INPUT	5	Program reads from current position	Standard input file for commands and definitions
OUTPUT	6	Program writes following current position	Standard file for printed output
SPDATA	8	Program repositions to beginning before each access	Alternate numeric input file
FEDATA	9	Program writes from current position	Card-image output file
SCRTCH	10	Not accessable to user	Temporary scratch file
T40141	None	Associated with graphic terminal	TEKTRONIX graphic input file
T40140	None	Associated with graphic terminal	TEKTRONIX graphic output file
TAPE61	61	Associated with graphic terminal	Alternate TEKTRONIX input file
TAPE62	62	Associated with graphic terminal	Alternate TEKTRONIX output file

SYSTEM CONTROL COMMANDS (CDC VERSION)

GPRIME users at DTNSRDC invoke the program with the following sequence of system commands. These commands are stored on a procedure file and invoke file assignments, housekeeping tasks, and the execution of the GPRIME program

automatically initiated by one command line. This file is referred to as a Cyber Control Language (CCL) Procedure and is listed in Figure 12. Most of the logic in this procedure is associated with the saving and restoring of the user's data base file, UMF. A new UMF file is created whenever the DBFILE parameter is set to "NEW" and a working copy is made of an existing UMF file whenever the DBFILE parameter is set to "OLD", the default. At the end of each session the working copy of the UMF is "cataloged" as a permanent file. The procedure does not remove any previous versions of the UMF from the permanent file system.

If an alternate input file is to be used, the name of that file must be associated with the INFILE parameter.

If the user chooses the permanent file name for his problem to be "XYZ PROBLEM" and the user's computer "ID" is "ABCD", the following sequences can be used to execute the program:

The procedure file is always attached by the command

```
ATTACH,GPRIME,ID=GPRM.
GPRIME,XYZPROBLEM,ABCD,NEW
```

for an initial run.

An equivalent form is

```
GPRIME,PF=XYZPROBLEM,ID=ABCD,DBFILE=NEW
```

Figure 12 - GPRIME CCL Procedure File

```
.PROC,GPRIME,PF=,ID=,DBFILE=OLD,INFILE=INPUT,END=$RETURN,XXX.$.
 DAY,OFF.
     COMMENT.  PF AND ID MUST IDENTIFY THE USERS DATA BASE FILE (UMF)
     COMMENT.  SET DBFILE TO "NEW" FOR INITIAL SESSIONS AND
     COMMENT.  TO "OLD" (DEFAULT) FOR RESTARTING.  "NEW" WILL BE ASSUMED
     COMMENT.  IF REFERENCED "OLD" FILE DOES NOT EXIST.
     COMMENT.  SET INFILE TO THE LOCAL FILE NAME OF THE PASSIVE INPUT
     COMMENT.  FILE, IF SUCH A FILE IS TO BE USED.
COMMENT.  SET END TO "DMP" TO OBTAIN A MEMORY DUMP OF GPRIME PROGRAM.
     COMMENT.
SET,R1=0.
```

Figure 12 (Continued)

```
RETURN,GPRIM,SMF,UMF,XMF.
REQUEST,UMF,*#PF.
IFE($PF$.EQ.$$.OR.$ID$.EQ.$$,NOPFN)
    SET,DSC=1.
    COMMENT.  MISSING #PF PARAM
    REVERT.
ENDIF,NOPFN.
ATTACH,GPRIM,GPRIM30,#ID=GPRM.
ATTACH,SMF,#ID=GPRM.
CONNECT,INPUT.
CONNECT, T40141.
CONNECT, T40140.
IFE(OT.EQ.TXO,SCRLBL)
    SCREEN,132.
ENDIF(SCRLBL)
IFE($DBFILE$.EQ.$OLD$,ATT)
    ATTACH,XMF,PF,#ID=ID.
    COPYBF,XMF,UMF.
    EXIT(U)
    RETURN,XMF.
ENDIF,ATT.
SET,R1=1.
DAY,ON.
GPRIM,INIFLE.
END,7500.
SET,R1=2.
CATALOG,UMF,PF,#ID=ID.
RETURN,T40141,T40140.
REVERT.
EXIT.
END,75000.
DAY,ON.
```

Figure 12 (Continued)

```
IFE(R1.EQ.1,DMPCAT)
    CATALOG,UMF,PF,#ID=ID.
ENDIF,DMPCAT.
RETURN,T40141,T40140,XMF.
REVERT.
EOF.
```

ACKNOWLEDGMENTS

Many people have contributed to the development of the GPRIME language. The authors would particularly like to acknowledge Suzanne Wybraniec for her role as programmer and program librarian for the project. Others who have contributed subroutines, program and documentation-building tools, editorial help, suggestions and encouragement include Donald Gignac, Melvin Haas, Richard Kazden, Edward Johnson, Jr., Randall Howard, Peter Roth, Hugh Tornebene. Special thanks go to those brave and persistent individuals who, as program users, have persevered through early versions of the language and the data generation programs, including Jane Figula, Melvyn Marcus, and Antonio Quezon.

REFERENCES

1. Golden, M.E., "Geometric Structural Modelling: A Promising Basis for Finite Element Analysis," published in "Trends in Computerized Structural Analysis and Synthesis," edited by A.K. Noor and H.G. McComb, Jr., Pergamon Press Ltd., Oxford (1978).

2. McKee, J.M. and R.J. Kazden, "GPRIME B-Spline Manipulation Package--Basic Mathematical Subroutines," DTNSRDC Report 77-0036 (Apr 1977).

3. Kazden, R.J., "Specifications for a Solid Finite Element Data Generator," DTNSRDC Departmental Report CMLD-78-04, March 1978.

4. Riesenfeld, R., "Application of B-Spline Approximation to Geometric Problems of Computer-Aided Design," University of Utah Computer Science Report UTEC-CSC-73-126, March 1973.

5. McKee, J.M., "B-Spline Functions as the Solution to Some Knotty Geometric Modeling Problems,"Proceedings of the Symposium on Computer Methods in Egineering, held at University of Southern California, Los Angeles, CA, Aug. 23-26, 1977.

6. deBoor, C., "A Practical Guide to Splines," Springer-Verlag, New York, 1978.

7. Schoenberg, I.J., "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions," Quart. Appl. Math., vol 4 (1946), pp. 45-99.

INITIAL DISTRIBUTION

Copies		Copies	
1	USA PICATINNY ARSENAL 1 B. Nagel	3	NSWC/White Oak 1 K22/R.J. Edwards 1 E22/E. Peizer 1 Tech Lib
1	USA BRL/Aberdeen 1 P. Deitz	2	NSWC/Dahlgren 1 K21/C. Blackman 1 Tech Lib
5	NRL 1 246/R. Perlut 1 2310/R. Shimkus 1 5840/R. Skop 1 7732/L. Turner 1 5836/W. Webbon	2	NUSC/New London 1 44/A. Carlson 1 Tech Lib
1	DNL	2	NUSC/Newport 1 3701/C. Curtis 1 Tech Lib
2	USNA 1 Dept. Math 1 Tech Lib	1	CNO 1 OP 901M/M. Golden
2	NADC 1 501/J. Heap 1 Tech Lib	2	NWC 1 32H2/J. Serpanos 1 Tech Lib
1	NATC/Patuxent 1 Don Louieos	1	NAVAIR 1 1154/G. Hand
3	NAVPGSCOL/Lib 1 G. Cantin 1 A. Shoenstadt 1 Tech Lib	3	NAVMAT 1 MAT 071 1 MAT 0714 1 MAT 064
1	NAVWARCOL	10	NAVSEA 1 SEA 03A 1 SEA 03R24 1 SEA 55Y 1 SEA 32113/J. Claffey 1 SEA 03R2/J. Gagorik 1 SEA 32323/R. McCarthy 1 SEA 03R2/J. Sejd 1 SEA 32133/W. Sandburg 1 SEA 05R15/H. Vanderveldt 1 SEA 03R3/P. Anklowitz
1	ROTC, MIT		
2	NCSC 1 751/P. Bishop 1 Tech Lib		
2	NOSC/San Diego 1 7132/L. McLeary 1 Tech Lib		
1	NPRDC 1 P311/D. Rahilly	1	NAVSHIPYD BREM/LIB

Copies

1 NAVSHIPYD CHASN/Lib

1 NAVSHIPYD MARE/Lib

1 NAVSHIPYD NORVA/Lib

1 NAVSHIPYD PEARL/Lib

1 NAVSHIPYD PHILA/Lib

1 NAVSHIPYD PTSMH/Lib

5 ONR
- 1 ONR 411
- 1 ONR 411 MA
- 1 ONR 432
- 1 ONR 437
- 1 ONR 474

12 DTIC

16 AFWAL/WPAFB, OH 45433
- 12 FIBRA/B. Groomes
- 1 FIBRA/V. Tishler
- 1 FIBE/D. Paul
- 1 J. Johnson
- 1 J. Folck

2 AFLC/Kelly AFB
- 1 J.L. Cotnam
- 1 M. Leo

1 NASA Headquarters
- 1 RWS/A.K. Amos

8 NASA Goddard SFC
- 1 J. Mason
- 1 R. Mitchell
- 1 C.E. Jackson, Jr.
- 1 L. Purves
- 3 G. Jones
- 1 Tech Lib

6 NASA Langley Research Center
- 1 R. Fulton
- 1 J. Rogers
- 1 P. Manos
- 1 G.L. Giles
- 1 J.C. Robinson
- 1 Tech Lib

Copies

2 NASA Lewis Research Center
- 1 MS 886-1/H.C. Kao
- 1 Tech Lib

2 NASA Flight Research Center
- 1 D. Hedgley
- 1 Tech Lib

2 NASA Marshall SFC
- 1 ED22/R. McComas
- 1 Tech Lib

3 NASA Ames Research Center
- 1 Mladen Chargin
- 1 Thomas Lasinski
- 1 Tech Lib

1 Sandia National Laboratories
- 1 5523/R. Koteras

1 Mr. Edwin N. Nilson, Manager
Pratt & Whitney Aircraft Group
Technical Management & Data Systems
400 Main Street
East Hartford, CT 06108

1 Dr. J.K. Lee
Dept. of Engineering Mechanics
Ohio State University
Boyd Laboratory
155 West Woodruff Avenue
Columbus, OH 43201

1 Mr. Brian J. McCartin-EB2C
Pratt & Whitney Aircraft Group
400 Main Street
East Hartford, CT 06108

1 Dr. Jeffrey Morgan, Vice Pres.
Universal Analytics, Inc.
7740 West Manchester Blvd.
Playa Del Rey, CA 90291

1 Dr. Jack C. Wiley
Deere & Company
Technical Center
Moine, IL 61265

Copies		Copies	
1	Mr. David Oswald CDM Division National Computer Systems 4401 West 76th Street Minneapolis, MN 55435	1	Mr. Bill Young Graphics Concepts, Inc. P.O. Box 14463 Columbus, OH 43202
1	R.P. Schmitz Sperry Support Service Huntsville Engineering Operations 1112 Church Street Huntsville, AL 35801	1	Dr. Lewis E. Hulbert Transportation & Structure Department Battelle Columbus Labs. 505 King Avenue Columbus, OH 43201
1	Dr. David Bushnell Lockheed Missiles & Space Co., Inc. Orgn. 52-33, Bldg. 205 3251 Hanover Street Palo Alto, CA 94304	1	Dr. George Allen Consultant, CAD/CAM McDonnell Douglas Automation Company 22500 S. Avalon Blvd. Carson, CA 90745
1	Dr. F. Hubert Ho The B.F. Goodrich Co. Research and Development Brecksville, OH 44141	1	Dr. Vangala S. Reddy Babcox and Wilcox 20 S. Van Buren Avenue Barberton, OH 44203
1	Dr. Joe Glouderman MacNeal-Schwendler Corp. 7442 N. Figureoa Street Los Angeles, CA 90041	1	Mike Bailey Mechanical Engineering Bldg. Purdue University West Lafayette, IN 47907
1	Mr. D.J. Tree Airesearch Manufacturing Co. of Arizona 111 South 34th Street P.O. Box 5217 Phoenix, AZ 85010	1	Z. Gabrijel Gas Dynamics Laboratories Aerospace Engineering Bldg. The University of Michigan Ann Arbor, MI 48109
1	Dr. Hussein A. Kamel Department of Aerospace and Mechanical Engineering University of Arizona Tucson, AZ 85721	1	Ronald Winter Bldg. 150B Tennessee Eastman Co. Kingsport, TN 37662
1	Dr. Albert L. Klosterman Vice President/Technical Dir. Structural Dynamics Research Corporation 2000 Eastman Drive Milford, OH 45150	1	Mr. C.J. Parekh Airesearch Manufacturing Co. of California 2525 West 190th Street Torrence, CA 90509

Copies		Copies	
1	Mr. C.H. Lee Lockheed Missiles & Space Co. Huntsville Research & Engineering Center P.O. Box 1103 Huntsville, AL 35807	1	Don Pan Bolt, Beranek, and Newman Union Station New London, CT 06320
1	Mr. S.W. Park Control Data Corporation 4201 Lexington Ave. North Arden Hills, MN 55112	1	Brenda Wilkey General Electric Co. Aircraft Engineering Group Mail Drop K190 Cincinnati, OH 45215
1	Mr. John N. Latta Science Applications, Inc. 1911 N. Ft. Meyer Drive Suite 1200 Arlington, VA 22209	1	Mr. Thomas Butler 932 Beaver Branch Circle Towson, MD 21204
1	Mr. M.J. Roche (B13-35) Grumman Aerospace Corp. Bethpage, NY 11714	1	H.D. Seamons MARC Analysis Research Corp. 260 Sheridan Palo Alto, CA 94306
1	Jim Cokonis Room M4018 VFSC General Electric Space Div. P.O. Box 8555 Philadelphia, PA 19101	1	Thomas P. Bligh Associate Professor of Mechanical Engineering Mass. Institute of Technology Cambridge, MA 02139
1	R. Elward Air Products & Chemicals P.O. Box 538 Allentown, PA 18105	1	Rudolph Brown Mail Stop 434 Westinghouse Electric Co. Systems Development Division P.O. Box 746 Baltimore, MD 21203
1	L.B. Stripling President and Engineering Consultant INTRATEC, Inc. 573 Pine Street Neptune Beach, FL 32233	1	Mr. Stephen M. Hollister Mail Stop C62 General Dynamics Corp. Electric Boat Division Eastern Point Road Groton, CT 06340
1	Dr. Richard L. Citerley Anamet Laboratories, Inc. 100 Industrial Way San Carlos, CA 94070	1	Dr. Bharat K. Soni Computational Fluid Dynamics Group, ED6 SVERDRUP Technology, Inc. AEDC Group, Mail Stop 500 Arnold Air Force Station, Tennessee 37389

Copies	
1	Dr. Oliver K.L. Wang SKF Industries P.O. Box 515 1100 First Avenue King of Prussia, PA 19406
1	Mr. Alfred Vachris RAVES Project Leader Grumman Aerospace Corp. B17-35 South Oyster Bay Road Bethpage, NY 11714
1	Robert Jackson Mail Drop K101 General Electric Co. I-75 Evendale, OH 45215
1	John S. Ma Structural Engineering Bldg. Office of Nuclear Reactor Regulation Nuclear Regulatory Commission Washington, D.C. 20555
1	Donald Scheiber Magnavox Company 1313 Production Road Fort Wayne, IN 46808

CENTER DISTRIBUTION

Copies	Code	Name
1	012.4	R. Stevens
1	11	W.M. Ellsworth
1	1182	Z.G. Wachnick
1	15	W.B. Morgan
1	1542	D.F. Thrasher
1	1568	L. Motter
1	16	H. Chaplin
1	17	W.W. Murray
1	1702	J. Corrado
1	1720.2	K. Hom
1	1720.3	R.F. Jones
1	1720.6	R.D. Rockwell
1	1730	A.B. Stavovy
1	1730.5	J.C. Adamchak
2	1750.2	B. Whang P. Roth
1	18	G.H. Gleissner
1	1802.1	H.J. Lugt
1	1805	E.H. Cuthill
1	1809.3	D. Harris
1	1820	A. Camara
1	1824	S. Berkowitz
1	1840	J.W. Schot
2	1843	H.J. Haussling M.B. Marquardt
3	1844	S.K. Dhir R. Kazden S. Wybraniec
50	1844	D.R. Wallace
100	1844	J.M. McKee
1	185	T. Corin
1	187	M. Zubkoff
1	189	G. Gray
1	1892.1	J. Strickland
1	1892.2	D. Sommer
1	19	M.M. Sevik
1	1966	J. Casper

Copies	Code	Name
1	27	W.C. Deitz
1	272.1	H.N. Urbach
1	2723	P. Hatchard
1	28	J.R. Belt
1	2822	W. Palko
2	2832	J. Dray
		T. Daugherty
1	29	F.H. Kendall, Jr.
10	5211.1	Reports Distribution
1	522.1	Unclassified Lib (C)
1	522.2	Unclassified Lib (A)
1	93	L. Marsh

DATE
ILMED
8